Springer Optimization and Its Applications 70

Michael Doumpos
Constantin Zopounidis
Panos M. Pardalos *Editors*

Financial Decision Making Using Computational Intelligence

Springer

Michael Doumpos • Constantin Zopounidis
Panos M. Pardalos
Editors

Financial Decision Making Using Computational Intelligence

Springer

Editors

Michael Doumpos
Department of Production
Engineering & Management
Technical University of Crete
University Campus
Chania, Greece

Constantin Zopounidis
Department of Production
Engineering & Management
Technical University of Crete
University Campus
Chania, Greece

Panos M. Pardalos
Center for Applied Optimization
ISE Department
University of Florida
Gainesville, FL, USA

and

Laboratory of Algorithms
and Technologies for Networks
Analysis (LATNA)
National Research University
Higher School of Economics
Moscow, Russia

ISSN 1931-6828
ISBN 978-1-4614-3772-7 ISBN 978-1-4614-3773-4 (eBook)
DOI 10.1007/978-1-4614-3773-4
Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2012942084

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Fast is fine, but accuracy is everything.
Xenophon (Greek historian, 431–350 BC)

Preface

The Context of Financial Decision Making

At the beginning of the twentieth century, finance was mainly a descriptive science focusing on institutional and legal aspects [9], but since then the field has experienced major transformations. These transformations began during the 1950s with the pioneering work of Harry Markowitz on portfolio theory [7] and intensified after then in the 1970s with the work of Black and Scholes on option pricing [1]. With these breakthroughs, the descriptive character of financial theory gradually progressed towards a more analytic one, which ultimately led to the engineering phase of finance by the late 1980s [8].

Todays, the financial industry is going through a tough period characterized by increasing uncertainties and risk challenges of diverse nature. The globalization of the business environment has generated several new opportunities and increased the pace of development for many financial instruments and innovations. In this context, investors have a wide range of options suitable for different investment policies. Managers of firms can use a variety of products for corporate financing and risk management, and policy makers face new challenges in choosing the best policies and measures for monitoring and controlling the markets in an effective way. Clearly, the decision-making process in the "new era" of finance is becoming more and more difficult, as all stakeholders seek effective operational decision-support tools to assess risks, assets, and funds, taking into particular consideration the new issues and concerns raised by the recent global crisis.

The financial decisions in firms and organization are of diverse nature, involving investment appraisal, risk management, asset valuation, capital budgeting, corporate financing, and performance evaluation among others. Depending on the problem at hand, different modeling and methodologies are applicable, such as:

- Regression models for risk analysis
- Econometric methods for time-series forecasting
- Stochastic calculus for asset pricing
- Optimization techniques for asset allocation

- Multiple criteria decision making for corporate performance evaluation and banking
- Simulation approaches for risk assessment

It is clear that, the financial decision modeling process cannot be contained within a set of specific methods and techniques. Instead, financial decision making usually requires the implementation of an interdisciplinary approach. Furthermore, the volume and complexity of financial data makes sophisticated computational procedures highly relevant. These computational procedures are not only restricted to efficient and scalable solution algorithms, but also provide new decision modeling and analysis capabilities. In this context, computational intelligence approaches have attracted considerable interest among researchers and practitioners.

Computational Intelligence in Financial Decision Making

Financial decisions are characterized by their highly complex and ill-structured nature, the increasing volume of data that needs to be analyzed, and the uncertainties involved in the decision context. Intelligent systems and techniques are well suited to this framework. Over the past decades, enormous progress has been made in the field of artificial intelligence in areas such as expert systems, knowledge-based systems, case-based reasoning, logic, data mining, and machine learning. Computational intelligence has emerged as a distinct sub-field of artificial intelligence involved with the study of adaptive mechanisms to enable intelligent behavior in complex and changing environments [4]. Typical computational intelligence paradigms include neural networks, support vector machines, and other machine learning algorithms, as well as evolutionary computation, nature-inspired computational methodologies, and fuzzy systems.

Computational intelligence methodologies have been successfully used in a variety of complex financial decision-making problems, providing researchers and practitioners with new solution algorithms for financial optimization problems as well as new decision modeling capabilities to handle the nonlinearities, uncertainties, and ambiguity involved in the financial decision-making process. While a comprehensive overview is beyond the scope of this introductory preface, some characteristic examples are given below.

- *Portfolio optimization*: The mean–variance model of Markowitz has set the basis of modern portfolio management. The basic model has been extended in various directions to take into consideration new risk measures (e.g., mean absolute deviation, value at risk, conditional value at risk, etc.), as well as several important factors in describing the investment policy of the decision maker (e.g., transaction costs, multiperiod horizons, cardinality constrained portfolios, round lots, index replication, etc.). While such extensions add much more realism in the modeling process, they also add considerable complexity, which is difficult to cope with through analytic optimization algorithms. Evolutionary algorithms are particularly useful in this context as they enable the efficient search of complex solution spaces. (See [6] for a comprehensive discussion with applications to portfolio optimization).

- *Financial time-series forecasting*: Time-series analysis and forecasting is a major topic within the field of finance, widely used to perform stock market predictions, forecast exchange rates, and interest rate modeling, among others. Traditional econometric forecasting approaches are usually based on linear models and impose specific assumptions about the relation between the variables. Computational intelligence methods, including neural network models, support vector machines, and other machine learning approaches, constitute a powerful alternative based on data-driven nonlinear modeling forms, which are free of statistical assumptions [5].
- *Trading systems*: The large volume of data that traders have to analyze in the financial markets has made algorithmic trading systems extremely popular. Such systems enable the analysis of the current state of the markets and the identification of short-term trends and patterns, based on technical analysis indicators. Given the high complexity of the financial markets and their chaotic behavior simple empirical trading strategies often fail to perform well, while the search for more complex trading rules is computationally difficult. However, evolutionary computation techniques such as genetic programming and genetic algorithms have been found particularly useful in this area (see for example [2]).
- *Credit scoring*: The development of credit risk rating models and systems is of major importance for financial institutions. Traditional statistical procedures such as discriminant analysis and logistic regression are commonly used for model development, but computational intelligence approaches have attracted much interest recently. For instance, neural networks, neuro-fuzzy systems, and other machine learning techniques have been successfully used to build credit scoring models with high predictive accuracy [3, 10]. Evolutionary methods and nature-inspired metaheuristics (e.g., swarm intelligence, ant colony optimization, etc.) have also been employed in this context to select the best set of predictor attributes and to construct hybrid scoring systems.

Outline of the Book

Aims and Scope

During the past decade, the research on the use of computational intelligence methods for financial decision making has grown considerably. The aim of this book is to present the recent advances made in this field covering both new methodological developments and new emerging application areas. On the methodological side, popular computational intelligence paradigms are presented in the book, such as machine learning, neural networks, evolutionary computation, Bayesian networks, Markov models, and fuzzy sets. The book also covers a wide range of applications related to financial decision making, financial modeling, risk management, and financial engineering, such as algorithmic trading, financial time-series analysis, asset pricing, portfolio management, auction market, and insurance services. Below we provide an outline to the contents of the book.

Organization

The book starts with a chapter written by Healy on the use of computational intelligence methods for building regression models in financial decision making. While such models are often used simply to obtain point estimates, the specification of intervals for their outputs is also very important in several contexts. Healy provides an exposition of different methods for estimating confidence and prediction intervals on outputs of neural network models, which are the best known and most successful computational intelligence technique for financial applications. The chapter also includes empirical comparative results on synthetic data, as well as data involving option pricing.

The second chapter by Chen, Shih, and Tai is concerned with the use of computational intelligence for building artificial traders who are capable of mimicking human traders' behavior. The authors present a genetic programming approach and propose a novel learning index which can better capture the state of human traders' learning dynamics. Computational results from double auction markets are given to illustrate the proposed framework.

The third chapter of the book, by Bozic, Chalup, and Seese, presents the contributions of intelligent systems for the analysis of market sentiment information. Analyzing the business and financial news in order to capture the sentiment of the market is an important part of the decision making with regard to the prediction of stock market movements. The chapter overviews the existing systems that perform automatic news analysis using text mining techniques in order to provide predictions for equity price movements on the financial markets, based on an estimated sentiment score. The chapter also presents some empirical results demonstrating how such a sentiment score relates to macroeconomic variables and closes with a presentation of an integrated system for performing sentiment analysis.

In the fourth chapter, Dunis, Laws, and Karathanasopoulos investigate the use of novel neural network architectures for stock market prediction. In particular, the chapter analyzes the trading performance of six different neural network architectures (including hybrid models) using data from the Greek stock market, and presents comparative results to traditional statistical and technical methods.

The fifth chapter, by Xylogiannopoulos, Karampelas, and Alhajj, focuses on data-mining techniques for financial time-series analysis involving exchange rates. The authors propose a new methodology that enables the extraction of valuable information about pattern periodicity, which can be used to interpret correlations among different market events or even to forecast future behavior. The proposed algorithmic procedure is of low computational complexity, thus allowing the analysis of extremely large time series.

The next two chapters are related to asset pricing. In particular, the chapter of Agapitos, O'Neill, and Brabazon involves the pricing of weather derivatives, which have become important financial instruments for protecting against commercial risks. The authors present a genetic programming approach to learn and forecast temperature profiles as part of the broader process of determining appropriate pricing model for weather derivatives. Furthermore, in order to improve the forecasting performance of the model an ensemble learning technique is employed combining the outputs of multiple models.

In the seventh chapter, Quintana, Luque, Valls, and Isasi develop a methodology for pricing initial public offerings (IPOs). IPOs have attracted considerable interest among researchers and practitioners due to the abnormal first-day trading returns that they often provide to investors. The authors review the contributions of computational intelligence in IPO pricing and develop a novel methodology based on an evolutionary system that combines multiple models in order to provide improved recommendations. The system is benchmarked against a set of well-known machine learning algorithms.

The following three chapters involve the applications of computational intelligence techniques to portfolio management and optimization. The first of these chapters, by Villa and Stella, develops a new framework for portfolio analysis and optimization based on Bayesian networks. The Bayesian networks' ability for efficient evidential reasoning is used to understand the behavior of an investment portfolio in different economic and financial scenarios. Thus, a portfolio optimization problem is formulated taking into account the investor's market views. The proposed framework is applied to data from the Dow Jones Euro Stoxx 50 Index.

The next chapter, by Bautin and Kalyagin, is concerned with the use of Markov chain models in portfolio optimization in the Russian stock market. First, the authors focus on multiperiod portfolio optimization and identify instabilities in the optimal portfolio related to the choice of the states of the Markov model. At the second stage, the analysis involves the structural changes on the market due to the financial crisis of 2008, leading to the detection of interesting dynamics due to the crisis.

In the next chapter, Vercher and Bermúdez present some possibilistic models for selecting portfolios considering different approaches for quantifying the uncertainty of future returns. In the proposed modeling approach, the uncertainties are taken into consideration using the theory of fuzzy sets, which provides a framework for the analysis of investment decisions under imperfect knowledge of future market behavior. The portfolio selection process is implemented through the formulation of a multi-objective optimization, which is solved through an evolutionary algorithm to find efficient portfolios that meet the investor's goals.

The book closes with the chapter by Corsaro, De Angelis, Marino, and Zanetti, which is concerned with the asset-liability management (ALM) process in insurance undertakings. In particular, the chapter focuses on some computational issues related to internal models, which are used by insurance undertakings as risk management systems developed to analyze the overall risk position and define the corresponding capital requirements.

Acknowledgements Sincere thanks must be expressed to all the authors who have devoted considerable time and effort to prepare excellent comprehensive works of high scientific quality and value. Without their help it would be impossible to prepare this book in line with the high standards that we have set from the very beginning of this project.

Chania, Greece Michael Doumpos
Chania, Greece Constantin Zopounidis
Gainesville, FL Panos M. Pardalos

References

1. F. Black, M. Scholes, The pricing of options and corporate liabilities. J. Polit. Econ. **81**, 659–674 (1973)
2. S.-H. Chen, T.-W. Kuo, K.-M. Hoi, Genetic programming and financial trading: how much about what we know, in *Handbook of Financial Engineering*, ed. by C. Zopounidis, M. Doumpos, P.M. Pardalos (Springer, New York, 2008), pp. 99–154
3. J.N. Crook, D.B. Edelman, L.C. Thomas, Recent developments in consumer credit risk assessment. Eur. J. Oper. Res. **183**, 1447–1465 (2007)
4. A.P. Engelbrecht, *Computational Intelligence: An Introduction* (Wiley, Chichester, 2002)
5. J. Kingdon, *Intelligent Systems and Financial Forecasting* (Springer, New York, 1997)
6. D. Maringer, *Portfolio Management with Heuristic Optimization* (Springer, Dordrecht, 2005)
7. H. Markowitz, *Portfolio Selection: Efficient Diversification of Investments* (Wiley, New York, 1959)
8. J.F. Marshall, M.P. Dorigan, Financial engineering: Information technology and its place in the new finance. Tech. Soc. **18**(2), 185–201 (1996)
9. R.C. Merton, Influence of mathematical models in finance on practice: past, present and future, in *Mathematical Models in Finance*, ed. by S.D. Howison, F.P. Kelly, P. Wilmott (Chapman and Hall, London, 1995), pp. 1–13
10. P. Ravi Kumar, V. Ravi, Bankruptcy prediction in banks and firms via statistical and intelligent techniques – A review. Eur. J. Oper. Res. **180**, 1–28 (2007)

Contents

Contributors

Alexandros Agapitos Financial Mathematics and Computation Research Cluster, Natural Computing Research and Applications Group, Complex and Adaptive Systems Laboratory, University College Dublin, Ireland, alexandros.agapitos@ucd.ie

Reda Alhajj Department of Computer Science, University of Calgary, Calgary, AB, Canada, alhajj@ucalgary.ca

Grigory A. Bautin National Research University Higher School of Economics, Lab LATNA, Russia, gbautin@hse.ru

José D. Bermúdez Department of Statistics and Operational Research, University of Valencia, Spain, bermudez@uv.es

Caslav Bozic Institute AIFB, Karlsruhe Institute of Technology, Karlsruhe, Germany, bozic@kit.edu

Anthony Brabazon Complex and Adaptive Systems Laboratory, University College Dublin, Ireland, anthony.brabazon@ucd.ie

Stephan Chalup School of Electrical Engineering and Computer Science, The University of Newcastle, Australia, stephan.chalup@newcastle.edu.au

Shu-Heng Chen AIECON Research Center, Department of Economics, National Chengchi University, Taiwan, chen.shuheng@gmail.com

Stefania Corsaro Dipartimento di Statistica e Matematica per la Ricerca Economica, Università degli Studi di Napoli "Parthenope", Napoli, Italy, corsaro@uniparthenope.it

Pasquale Luigi De Angelis Dipartimento di Statistica e Matematica per la Ricerca Economica, Università degli Studi di Napoli "Parthenope", Napoli, Italy, deangelis@uniparthenope.it

Christian L. Dunis Liverpool John Moores University, Liverpool, UK, C.Dunis@ljmu.ac.uk

Jerome V. Healy UEL Royal Docks Business School, University of East London, London, UK, j.healy@uel.ac.uk

Pedro Isasi Universidad Carlos III de Madrid, Madrid, Spain, isasi@ia.uc3m.es

Valery A. Kalyagin National Research University Higher School of Economics, Lab LATNA, Russia, vkalyagin@hse.ru

Panagiotis Karampelas Hellenic American University, Manchester, NH, USA, pkarampelas@hauniv.us

Andreas Karathanasopoulos London Metropolitan University, London, UK, a.karathanasopoulos@londonmet.ac.uk

Jason Laws University of Liverpool Management School, Liverpool, UK, J.Laws@liverpool.ac.uk

Cristobal Luque Universidad Carlos III de Madrid, Madrid, Spain, cluqueac@gmail.com

Zelda Marino Dipartimento di Statistica e Matematica per la Ricerca Economica, Università degli Studi di Napoli "Parthenope", Napoli, Italy, marino@uniparthenope.it

Michael O'Neill Financial Mathematics and Computation Research Cluster, Natural Computing Research and Applications Group, Complex and Adaptive Systems Laboratory, University College Dublin, Ireland, m.oneill@ucd.ie

David Quintana Universidad Carlos III de Madrid, Madrid, Spain, dquintan@inf.uc3m.es

Detlef Seese Institute AIFB, Karlsruhe Institute of Technology, Karlsruhe, Germany, detlef.seese@kit.edu

Kuo-Chuan Shih AIECON Research Center, Department of Economics, National Chengchi University, Taiwan, melvinshih@gmail.com

Fabio Stella University of Milano – Bicocca, Milano, Italy, stella@disco.unimib.it

Chung-Ching Tai Department of Economics, Tunghai University, Taiwan, chungching.tai@gmail.com

Jose Maria Valls Universidad Carlos III de Madrid, Madrid, Spain, jvalls@inf.uc3m.es

Enriqueta Vercher Department of Statistics and Operational Research, University of Valencia, Burjassot, Spain, vercher@uv.es

Simone Villa University of Milano - Bicocca, 20126 Milano, Italy

Saint George Capital Management, Lugano, Switzerland, simone.villa@sgcm.ch

Konstantinos F. Xylogiannopoulos Hellenic American University, Manchester, NH, USA, kostasfx@yahoo.gr

Paolo Zanetti Dipartimento di Statistica e Matematica per la Ricerca Economica, Università degli Studi di Napoli “Parthenope”, Napoli, Italy, zanetti@uniparthenope.it

Chapter 1
Statistically Principled Application of Computational Intelligence Techniques for Finance

Jerome V. Healy

Abstract Computational techniques for regression have been widely applied to asset pricing, return forecasting, volatility forecasting, credit risk assessment, and value at risk estimation, among other tasks. Determining probabilistic bounds on results is essential in these contexts. This chapter provides an exposition of methods for estimating confidence and prediction intervals on outputs, for computational intelligence tools used for data modelling. The exposition focuses on neural nets as exemplars. However, the techniques and theory outlined apply to any equivalent computational intelligence technique used for regression. A recently developed robust method of computing prediction intervals, appropriate to any such regression technique of sufficient generality, is described.

1.1 Computational Intelligence Techniques in Finance

In the past 2 decades there has been increasing interest among both practitioners and scholars in using techniques from the field of computational intelligence for modelling to support financial decisions. Recent examples documented in the literature include: the use of radial basis functions [8], projection pursuit regression [5], multilayer perceptrons (MLPs) [22], genetic algorithms [1], and support vector machines [27]. It has been shown [7, 11] that these techniques possess universal approximation properties and can approximate arbitrarily closely virtually any linear or non-linear function. Techniques of these types are now frequently used for asset pricing, return forecasting, volatility forecasting, credit risk assessment, and Value at Risk (VaR) estimation, among other tasks. Determining probabilistic bounds on the resulting estimates and predictions is essential in such applications.

J.V. Healy (✉)
UEL Royal Docks Business School, University of East London, Docklands Campus, 4-6 University Way, London E16 2RD, UK
e-mail: j.healy@uel.ac.uk

Consider the task of forecasting VaR, in support of decisions on regulatory capital requirements. The VaR for a portfolio is the largest expected loss likely to occur with defined probability p, over a defined horizon $T - t = \tau$. VaR is thus the upper bound of a (single sided) prediction interval of expected portfolio losses in statistical terms. Formally, if we wish to determine the 95% VaR of a portfolio V over some horizon τ, where V_t is the value of the portfolio today and V_T is the expected value of the portfolio at the end of period τ, then we define $\text{Prob}(L_\tau \leq \text{VaR}) = 95\%$. Thus L_τ is the maximum expected loss in the period and will have a positive value (negative values represent a gain). The 95% VaR for L_τ is therefore the upper bound of the 95% prediction interval for the conditional probability distribution of L_τ. If we wish to apply computational intelligence to forecast the VaR for L_τ we thus require a method for estimating prediction intervals for the chosen technique.

The computational intelligence techniques referred to above, and others in this class, such as decision trees, or k-nearest neighbour classifiers, may appear superficially disparate. However, Maruyama et al. [18] have demonstrated the underlying equivalence among a broad class of such techniques, including those mentioned. Given the wide variety of available techniques and their essential equivalence, the exposition in this chapter necessarily focuses on what is arguably the best known and most successful computational intelligence technique for financial applications. Namely, the form of neural network termed a multilayer perceptron (MLP). However, it is shown that under mild conditions, the theory and methods discussed are applicable to all such techniques used for regression.

An MLP with a single hidden layer can approximate arbitrarily closely virtually any linear or non-linear continuous function [11]. MLPs are also twice differentiable, so first- and second-order partial derivatives of the network with respect to its inputs are obtainable [12]. These characteristics make MLPs well suited to applications in finance.

1.2 Theory

To define terms and develop notation the theory relating to confidence and prediction intervals applied to regression is first reviewed. A discussion of how this theory applies to computational intelligence techniques then follows.

1.2.1 Confidence and Prediction Intervals

Regression is the name given to the family of statistical techniques used to model the relationship between a *response* (or *dependent*) *variable* y and a vector $\mathbf{x}$ of *explanatory* (or *independent*) *variables*, the *regressors*. In the learning network literature the terminology *targets* for response variables and *inputs* for regressors, with the vector $\mathbf{x}$ termed the *input vector*, is used. The following discussion adopts this usage. In regression, it is assumed there is a relationship between the target y and the input vector $\mathbf{x}$. Equation (1.1) shows a possible form this may take.

$$y = \mu_y(\mathbf{x}) + \varepsilon. \tag{1.1}$$

The relationship has both stochastic and deterministic components. Here $\varepsilon \sim N(0,\sigma^2)$ is a normally distributed random error. The stochastic component consists of the resulting random fluctuation of y about its mean $\mu_y(\mathbf{x})$. The deterministic component is the function relating $\mu_y(\mathbf{x})$ and $\mathbf{x}$. Suppose that the true but unknown function relating $\mu_y(\mathbf{x})$ and $\mathbf{x}$ is given by

$$\mu_y(\mathbf{x}) = f(\mathbf{x};\boldsymbol{\beta}), \tag{1.2}$$

where $\boldsymbol{\beta}$ is a set of parameters. Regression attempts to model this relationship by estimating the parameter values from the data set. To achieve this, the values of $\boldsymbol{\beta}$ are adjusted under the assumption that f is the true function, giving

$$\hat{\mu}_y(\mathbf{x};\hat{\boldsymbol{\beta}}) = f(\mathbf{x};\hat{\boldsymbol{\beta}}), \tag{1.3}$$

where a hat denotes an estimated value. The right-hand side of (1.3) is termed a *regression function*. If $\hat{\mu}_y(\mathbf{x};\hat{\boldsymbol{\beta}})$ is estimated from a finite sample $\mathbf{S}$, $\{(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),\ldots,(\mathbf{x}_n,y_n) \in \mathbf{S}\}$, sampling variation in $\mathbf{S}$ will result in variation in $\hat{\boldsymbol{\beta}}$ and hence variation in $\hat{\mu}_y(\mathbf{x};\hat{\boldsymbol{\beta}})$. It follows $\hat{\mu}_y(\mathbf{x}_0;\hat{\boldsymbol{\beta}})$ has a sampling distribution about $\mu_y(\mathbf{x}_0)$, where $\mathbf{x}_0$ is a particular value of $\mathbf{x}$. A 95% confidence interval for $\mu_y(\mathbf{x}_0)$ is an interval $[\lambda_L(\mathbf{S},\mathbf{x}_0),\lambda_U(\mathbf{S},\mathbf{x}_0)]$ about $\hat{\mu}_y(\mathbf{x}_0;\hat{\boldsymbol{\beta}})$ such that $\mu_y(\mathbf{x}_0)$ is within the interval in 95% of cases. A 95% prediction interval is an interval $[\lambda_L(\mathbf{S},\mathbf{x}_0),\lambda_U(\mathbf{S},\mathbf{x}_0)]$ about $\hat{\mu}_y(\mathbf{x}_0;\hat{\boldsymbol{\beta}})$ such that the unknown value y_0 associated with $\mathbf{x}_0$ is within the interval in 95% of cases. As an example consider the univariate (OLS) regression of y on x for a sample $\mathbf{S}$, $\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n) \in \mathbf{S}\}$. The 95% confidence interval for $\mu_y(\mathbf{x}_0)$ is given by

$$\hat{\mu}_y(x_0;\hat{\beta}) \pm t_{0.025(n-2)} \left(s_y \sqrt{\frac{1}{n} + \frac{(x_0-\bar{x})^2}{\sum_{i=1}^{n}(x_i^2)}} \right) \tag{1.4}$$

and the 95% prediction interval by

$$\hat{\mu}_y(x_0;\hat{\beta}) \pm t_{0.025(n-2)} \left(s_y \sqrt{\frac{1}{n} + \frac{(x_0-\bar{x})^2}{\sum_{i=1}^{n}(x_i^2)} + 1} \right), \tag{1.5}$$

where s_y is the standard deviation of the y values and $\bar{x}$ is the mean of the x values [23]. In (1.4) and (1.5), s_y is given by

$$s_y = \sqrt{\frac{\sum e_i^2}{n-2}}, \tag{1.6}$$

where $e = (\hat{\mu}_y(x_i;\hat{\beta}) - y_i)$ are the residuals. This follows from the classical assumptions for OLS regression under which $\mathrm{Var}(y_i) = \mathrm{Var}(\varepsilon_i) = \sigma^2$. An unbiased estimate of σ and hence of the standard deviation of the y_i is given by (1.6). Equations (1.4) and (1.5) can be generalised to the multivariate case for OLS regression to obtain a $(1-\alpha)100\%$ confidence interval for $\mu_y(\mathbf{x}_0)$. In matrix notation this is

$$\hat{\mu}_y(\mathbf{x}_0;\hat{\boldsymbol{\beta}}) \pm t_{(\alpha/2)(n-k-1)}\left(\sqrt{\mathbf{x}_0^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_0 s_y^2}\right) \tag{1.7}$$

and a $(1-\alpha)100\%$ prediction interval is

$$\hat{\mu}_y(\mathbf{x}_0;\hat{\boldsymbol{\beta}}) \pm t_{(\alpha/2)(n-k-1)}\left(\sqrt{(1+\mathbf{x}_0^\top(\mathbf{X}^\top\mathbf{X})^{-1})\mathbf{x}_0 s_y^2}\right). \tag{1.8}$$

In (1.7) and (1.8) $\mathbf{x}_0$ is a $k \times 1$ column vector of inputs and $\mathbf{X}$ is a $n \times k$ matrix, containing first a column of ones, and then the $k \times 1$ values of each of the n row vectors $\mathbf{x}_i^\top$. The scalar s_y^2 is the mean squared residual (MSR). It is an unbiased estimate of σ^2 and hence of the variance of the y_i. For (1.7) and (1.8), s_y^2 is given by

$$\mathrm{MSR} = s_y^2 = \frac{\mathbf{Y}^\top\mathbf{Y} - \hat{\boldsymbol{\beta}}^\top\mathbf{X}^\top\mathbf{Y}}{n-k-1}. \tag{1.9}$$

In (1.9), $\mathbf{X}$ is as previously defined, $\mathbf{Y}$ is an $n \times 1$ vector of target values, and $\hat{\boldsymbol{\beta}}$ is a $k \times 1$ vector of estimated parameters given by $\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}$. If the $\mathbf{x}$ values in (1.4)–(1.9) are continuously valued over the interval $[x_1, x_2]$, $x_2 > x_1$ a continuous *confidence band* and *prediction band* are obtained. From (1.2) $f(\mathbf{x};\boldsymbol{\beta})$ is the true but unknown function relating $\mu_y(\mathbf{x})$ and x (the *true regression*). Equation (1.3) $\hat{\mu}_y(\mathbf{x};\boldsymbol{\beta}) = f(\mathbf{x};\boldsymbol{\beta})$ is an estimate of this regression. It follows that the confidence intervals (1.4) and (1.7) are for the true regression functions. The prediction intervals (1.5) and (1.8) are for predicted values associated with a new unseen input. The relationship between confidence intervals and prediction intervals can be understood by considering the following equation:

$$[y - f(\mathbf{x};\hat{\boldsymbol{\beta}})] = [f(\mathbf{x};\boldsymbol{\beta}) - f(\mathbf{x};\hat{\boldsymbol{\beta}})] + \varepsilon(\mathbf{x}) \tag{1.10}$$

Prediction intervals are concerned with the left-hand side of this equation, the difference between the target value and its predicted value from the estimated regression function. The left-hand term decomposes into the two right-hand terms. Confidence intervals are concerned with the first of these, the difference between the true and estimated regression functions. This difference is determined by the parameter difference $\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}$. The remaining term on the right-hand side is the (sample dependent) random noise term. Neither the noise or the true parameters $\boldsymbol{\beta}$ can be directly observed. However, the variance of the noise can be estimated as $\widehat{\mathrm{Var}}(\varepsilon) = s^2$ and confidence intervals constructed for the true $\boldsymbol{\beta}$.

1.2.2 Application to Computational Intelligence Techniques

The theory presented in the previous section can be applied where the right-hand side of (1.3), the regression function, is a neural net or other computational intelligence technique used for regression. Existing methods are discussed in this section. The discussion uses an MLP with one hidden layer as an exemplar. In this case, from (1.3),

$$\hat{\mu}_y(\mathbf{x};\hat{\boldsymbol{\Omega}}) = \boldsymbol{\Theta}\left(\sum_{h=1}^{H} w_{lh}\Phi_h\left(\sum_{k=1}^{K} w_{hk}x_k + \omega_h\right) + \omega_l\right). \tag{1.11}$$

In (1.11), the MLP consists of one layer of K input nodes $x_1,\ldots,x_K$, a layer of l output nodes, and H hidden layer nodes [3]. The functions $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$ are termed activation functions. For the hidden layer, $\boldsymbol{\Phi}$ is usually a sigmoid function such as the logistic function or the hyperbolic tangent function. For a continuously valued target variable, the output functions $\boldsymbol{\Theta}$ are usually linear and may be the identity. The w are referred to as the weights and the ω are constant intercept terms known as biases. The set of estimated weights and biases is denoted by $\{\hat{\boldsymbol{\Omega}}(w_1,\ldots,w_{KH+Hl},\omega_1,\ldots,\omega_{H+l}) \in \hat{\boldsymbol{\Omega}}\}$. Unlike classical NLLS regression, in neural nets, the form of non-linearity involved is unknown. The error surface can have many local minima, and a closed form solution for the global minimum is not generally possible. The network is fitted ("trained" in the learning networks literature) by searching a weight space to select $\hat{\boldsymbol{\Omega}}$ to minimise a cost function. This is done using a search algorithm such as gradient descent, conjugate gradient, or quasi-Newton [3]. These methods require initialisation of the vector $\boldsymbol{\Omega}$ with a set of small random values. The vector $\hat{\boldsymbol{\Omega}}$ that minimises the cost function is then iteratively estimated. Neural nets can over-fit data. To prevent over-fitting, training is terminated when the error function is minimised on a portion of the data withheld to form a separate validation data set (*early stopping*). Alternatively, some form of regularisation (*weighting*) can be used.

When the cost function used for (1.11) is the sum of squared errors, some activation functions are non-linear (the usual case), and early stopping is used to prevent over-fitting, the optimisation is effectively a non-linear least squares regression (NLLS). The theory for estimating standard errors for non-linear regression is then directly applicable. It has been shown [15] that asymptotically correct standard errors can be estimated in this case. The estimation of standard errors for MLPs using the Delta Method [21] and Sandwich Method [13] for non-linear regression is outlined in Sect. 1.2.2.1. Alternative bootstrap and Bayesian approaches are outlined in Sect. 1.2.2.2.

1.2.2.1 The Delta and Sandwich Methods

Consider a sample $\mathbf{S}$, $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n) \in \mathbf{S}\}$ where y represents scalar targets and $\mathbf{x}$ a vector of k inputs. Suppose the true relationship between the targets y and the input vectors $\mathbf{x}$ takes the form

$$y_i = f(\mathbf{x}_i; \boldsymbol{\beta}) + \varepsilon_i \quad i = 1, \ldots, n. \tag{1.12}$$

The data are modelled by the regression equation (1.3) where the right-hand side is an MLP or equivalent computational intelligence technique. Thus the vector of parameters $\boldsymbol{\beta}$ is replaced by the set of weights and biases $\boldsymbol{\Omega}$, and

$$\hat{\mu}_y(\mathbf{x}_i; \hat{\boldsymbol{\Omega}}) = f(\mathbf{x}_i; \hat{\boldsymbol{\Omega}}) + e_i \quad i = 1, \ldots, n. \tag{1.13}$$

If n is large enough so that $\hat{\boldsymbol{\Omega}} \approx \boldsymbol{\Omega}$, a local linear approximation of the network about $\mathbf{x}_0 = \mathbf{x}$, where $\mathbf{x}_0$ is a particular value of $\mathbf{x}$, can be obtained and the procedures for multivariate linear regression applied. From a first-order Taylor series,

$$\hat{\mu}_y(\mathbf{x}_0; \hat{\boldsymbol{\Omega}}) = f(\mathbf{x}_0; \hat{\boldsymbol{\Omega}}) \approx f(\mathbf{x}_0; \boldsymbol{\Omega}) + \mathbf{g}_0^\top \Delta\hat{\boldsymbol{\Omega}}, \tag{1.14}$$

where $\Delta\hat{\boldsymbol{\Omega}} = (\hat{\boldsymbol{\Omega}} - \boldsymbol{\Omega})$ and $\mathbf{g}_0 = \mathrm{d}f(\mathbf{x}_0; \boldsymbol{\Omega})/\mathrm{d}\boldsymbol{\Omega}$.

Substituting this approximation into the least squares cost function gives the following expression for the sum of squared residuals:

$$\mathrm{SSR} \approx \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i; \boldsymbol{\Omega}) - \mathbf{g}_i^\top \Delta\hat{\boldsymbol{\Omega}})^2 \approx \sum_{i=1}^{n} (e_i - \mathbf{g}_i^\top \Delta\hat{\boldsymbol{\Omega}})^2. \tag{1.15}$$

Rewriting (1.15) in matrix notation gives

$$\mathrm{SSR} \approx (\mathbf{e} - \mathbf{G}\Delta\hat{\boldsymbol{\Omega}})^\top (\mathbf{e} - \mathbf{G}\Delta\hat{\boldsymbol{\Omega}}), \tag{1.16}$$

where $\mathbf{G}$ is an $n \times k$ matrix whose ith row is the vector $\mathbf{g}_i^\top$ and $\mathbf{e}$ is an $n \times 1$ vector of errors. Setting the derivatives of (1.16) with respect to $\hat{\boldsymbol{\Omega}}$ equal to zero and solving the resulting equations to minimise the SSR gives

$$\Delta\hat{\boldsymbol{\Omega}} \approx (\mathbf{G}^\top \mathbf{G})^{-1} \mathbf{G}^\top \mathbf{e}. \tag{1.17}$$

Different samples will generate different weight vectors according to (1.17). The variance–covariance matrix of the weights is (the derivation is not given here)

$$\widehat{\mathrm{Var}}(\Delta\hat{\boldsymbol{\Omega}}) \approx s_y^2 (\mathbf{G}^\top \mathbf{G})^{-1} \approx \mathbf{H}^{-1}, \tag{1.18}$$

where $\mathbf{H}^{-1}$ is the outer product approximation to the Hessian matrix of second-order partial derivatives of the cost function (1.16) with respect to each of the elements of the weight matrix $\hat{\boldsymbol{\Omega}}$. From (1.14) and (1.18) the standard error of the regression function can now be defined as

$$\widehat{\mathrm{SE}}(f(\mathbf{x}_i;\boldsymbol{\Omega})) \approx \left(\sqrt{\mathbf{g}_i^{\top}(\mathbf{G}^{\top}\mathbf{G})^{-1}\mathbf{g}_i s_y^2}\right) \approx \left(\sqrt{\mathbf{g}_i^{\top}\mathbf{H}^{-1}\mathbf{g}_i}\right). \tag{1.19}$$

Using (1.19), a $(1-\alpha)100\%$ confidence interval for $\mu_y(\mathbf{x}_0)$ is given by

$$\hat{\mu}_y(\mathbf{x}_0^{\top};\hat{\boldsymbol{\Omega}}) \pm t_{(\alpha/2)(n-k-1)}\left(\sqrt{\mathbf{g}_0^{\top}(\mathbf{G}^{\top}\mathbf{G})^{-1}\mathbf{g}_0 s_y^2}\right) \tag{1.20}$$

and a $(1-\alpha)100\%$ prediction interval is

$$\hat{\mu}_y(\mathbf{x}_0^{\top};\hat{\boldsymbol{\Omega}}) \pm t_{(\alpha/2)(n-k-1)}\left(\sqrt{(1+\mathbf{g}_0^{\top}(\mathbf{G}^{\top}\mathbf{G})^{-1}\mathbf{g}_0)s_y^2}\right). \tag{1.21}$$

Equations (1.20) and (1.21) have the same form as (1.7) and (1.8) in Sect. 1.2.1, with vector $\mathbf{g}$ substituted for $\mathbf{x}$ and matrix $\mathbf{G}$ substituted for $\mathbf{X}$. The s_y^2 term is $\mathrm{SSR}/(n-k-1)$, the MSR obtained using (1.16) in the numerator. Note that k here is the number of effective weights and biases.

If regularisation as described by Bishop [3] rather than early stopping is used to prevent over-fitting, the standard error given in (1.19) must be replaced by

$$\widehat{\mathrm{SE}}(f(\mathbf{x}_i;\boldsymbol{\Omega})) \approx \left(\sqrt{\mathbf{g}_i^{\top}(\mathbf{H}^{-1}+2\lambda)\mathbf{g}_i}\right), \tag{1.22}$$

where the cost function to be minimised is

$$\mathrm{SSR} + \lambda \sum_{i=1} n w_i^2 \tag{1.23}$$

and the λ term in (1.23) is a penalty term used to induce weight decay. The w_i are the weights from (1.11).

The value of k, the degrees of freedom to use in (1.20) and (1.21), is somewhat problematic. When training is stopped before convergence, by regularisation or early stopping, some weights and biases may be ineffective. An architecture selection algorithm should be applied, to ensure there are no redundant hidden layer nodes, thereby minimising or eliminating redundant weights. Assuming there are no redundant weights or hidden units in the network, $k = 1 + H(q+2)$, where q is the number of input variables and H the number of hidden layer nodes. Otherwise, this is an upper bound. In either case, for large data sets (e.g. 7,000 + observations), with $k > 50$, use of the above should have minimal effect on estimation accuracy.

An important assumption underlying the OLS and delta method estimators of standard error concerns the variance of the noise ε associated with the true regression. This is assumed $\varepsilon \sim N(0,\sigma^2)$ with constant variance σ^2. The presence of heteroskedasticity will result in estimated standard errors that are biased. The sandwich estimator [13] provides a method of dealing with this problem. The sandwich estimator is obtained by replacing the variance–covariance matrix of weights given by (1.18) with

$$\widehat{\mathrm{Var}}_{\mathrm{Sand}}(\Delta\hat{\boldsymbol{\Omega}}) \approx \frac{n\left[(\mathbf{G}^{\top}\mathbf{G})^{-1}\mathbf{G}^{\top}(\sum_{i=1}^{n}\mathbf{g}_i\mathbf{g}_i^{\top}e_i^2)\mathbf{G}(\mathbf{G}^{\top}\mathbf{G})^{-1}\right]}{(n-k)}. \tag{1.24}$$

Substituting (1.24) into (1.19) in place of (1.18) gives

$$\widehat{\mathrm{SE}}_{\mathrm{Sand}}(f(\mathbf{x}_i;\boldsymbol{\Omega})) \approx \left(\sqrt{\mathbf{g}_i^{\top}\left(\widehat{\mathrm{Var}}_{\mathrm{Sand}}(\Delta\hat{\boldsymbol{\Omega}})\right)\mathbf{g}_i}\right). \tag{1.25}$$

Equation (1.24) yields asymptotically consistent variance–covariance matrix estimates without making distributional assumptions, even if the assumed model underlying the parameter estimates is incorrect. Because of these desirable properties, the sandwich estimator is also termed the *robust covariance matrix estimator* or the *empirical covariance estimator*.

1.2.2.2 Bootstrap and Bayesian Approaches

Re-sampling methods provide an alternative to the delta method for calculating standard errors and statistical intervals for neural nets. The *bootstrap method* [6] is a computer-based technique based on re-sampling that can provide confidence intervals for any population parameter estimate. In the context of regression, two forms of bootstrap are possible. The first of these is the bootstrap pair method. Consider a sample $\mathbf{S}$, $\{(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),\ldots,(\mathbf{x}_n,y_n)\in\mathbf{S}\}$, where y represents scalar targets and $\mathbf{x}$ is a vector of k inputs. A *bootstrap sample* is a sample $\mathbf{S}^{\mathrm{Boot}}$, $\{(\mathbf{x}_i,y_i)\in\mathbf{S}^{\mathrm{Boot}}, i=1,\ldots,n\}$, consisting of n pairs of $(\mathbf{x}_i,y_i)$ drawn randomly (with replacement) from $\mathbf{S}$. This means that some $(\mathbf{x}_i,y_i)$ may appear more than once in $\mathbf{S}^{\mathrm{Boot}}$ while others may not appear at all. The bootstrap estimate of the standard error of the true regression function, which is a function of the set of inputs $\mathbf{X}$, is given by

$$\widehat{\mathrm{SE}}_{\mathrm{Boot}}(f(\mathbf{X};\boldsymbol{\Omega})) \approx \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left[\hat{\mu}_y(\mathbf{X}_b;\hat{\boldsymbol{\Omega}})-\hat{\mu}_{y,\mathrm{Boot}}(\mathbf{X})\right]^2}. \tag{1.26}$$

In (1.26), $\hat{\mu}_y(\mathbf{X}_b;\hat{\boldsymbol{\Omega}})$ is the network trained on the bth bootstrap sample $\mathbf{S}_b^{\mathrm{Boot}}$, where there are a total of B bootstrap samples, with typical values $20 < B < 200$.

The bootstrap estimate of the mean of the target values $\hat{\mu}_{y,\mathrm{Boot}}(\mathbf{X})$, termed a *bagged* estimate in the neural net literature, is given by the mean of the ensemble of B networks:

$$\hat{\mu}_{y,\mathrm{Boot}}(\mathbf{X}) \approx \frac{1}{B}\sum_{b=1}^{B} \hat{\mu}_y(\mathbf{X}_b; \hat{\boldsymbol{\Omega}}). \tag{1.27}$$

In (1.26) and (1.27), $\mathbf{X}$ is an $n \times 1$ vector of the $\mathbf{x}_i$, that is, an $n \times k$ matrix of x values. Using (1.26), a $(1-\alpha)100\%$ bootstrap confidence interval for $\mu_y(\mathbf{X})$ is given by [10]:

$$\hat{\mu}_{y,\mathrm{Boot}}(\mathbf{X}) \pm t_{(\alpha/2)B}\left(\widehat{\mathrm{SE}}_{\mathrm{Boot}}(f(\mathbf{X};\boldsymbol{\Omega}))\right). \tag{1.28}$$

In the *bootstrap residuals* method the residuals from $\hat{\mu}_y(\mathbf{X};\hat{\boldsymbol{\Omega}})$ trained on a sample $\mathbf{S}$, defined as before, are re-sampled rather than the training sample itself. Suppose $\mathbf{E}$, $\{(e_1, e_2, \ldots, e_n) \in \mathbf{E}\}$ is a set of residuals from $\hat{\mu}_y(\mathbf{X};\hat{\boldsymbol{\Omega}})$ trained on sample $\mathbf{S}$. A *bootstrap residual sample* is a sample $\mathbf{E}^{\mathrm{Boot}}$, $\{e_i \in \mathbf{E}^{\mathrm{Boot}}, i = 1, \ldots, n\}$ consisting of n samples e_i drawn randomly (with replacement) from $\mathbf{E}$. The bootstrap residual estimate of the standard error of the true regression function is given by

$$\widehat{\mathrm{SE}}_{\mathrm{BootR}}(f(\mathbf{X};\boldsymbol{\Omega})) \approx \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left[\hat{\mu}_y^r(\mathbf{X};\hat{\boldsymbol{\Omega}}^b) - \hat{\mu}_{y,\mathrm{BootR}}(\mathbf{X})\right]^2}. \tag{1.29}$$

In (1.29), $\hat{\mu}_y^r(\mathbf{X};\hat{\boldsymbol{\Omega}}^b)$ is the NN trained on the bth bootstrap residual sample $\mathbf{E}_b^{\mathrm{Boot}}$, where there are a total of B bootstrap residual samples, with typical values $20 < B < 200$. The target for $\hat{\mu}_y^r(\mathbf{X};\hat{\boldsymbol{\Omega}}^b)$ is $(\mathbf{E}_b^{\mathrm{Boot}} + \hat{\mu}_y^r(\mathbf{X};\hat{\boldsymbol{\Omega}}^b))$. The bootstrap residual estimate of the mean of the target values, $\hat{\mu}_{y,\mathrm{BootR}}(\mathbf{X})$, is given by the mean of the ensemble of B networks:

$$\hat{\mu}_{y,\mathrm{BootR}}(\mathbf{X}) \approx \frac{1}{B}\sum_{b=1}^{B}\hat{\mu}_y^r(\mathbf{X};\hat{\boldsymbol{\Omega}}^b). \tag{1.30}$$

Using (1.29), a $(1-\alpha)100\%$ bootstrap confidence interval for $\mu_y(\mathbf{X})$ is

$$\hat{\mu}_{y,\mathrm{BootR}}(\mathbf{X}) \pm t_{(\alpha/2)B}\left(\widehat{\mathrm{SE}}_{\mathrm{BootR}}(f(\mathbf{X};\boldsymbol{\Omega}))\right). \tag{1.31}$$

The bootstrap residuals method has the advantage that the same sample $\mathbf{S}$ is the source of the inputs $\mathbf{X}$ for all B networks that must be trained. This may be an advantage in some experimental situations. On the other hand, it is model specific and not as robust to over-fitting or misspecification as the bootstrap pair method.

The delta, sandwich, and bootstrap estimators of standard errors are based on the maximum likelihood framework. Bayesian statistics provides a different approach. In classical "frequentist" statistics, inferences about the parameters of a population

$\mathbf{P}$ are based entirely on sample statistics from sample(s) $\mathbf{S}$ drawn randomly from $\mathbf{P}$. Bayesian statistics, in contrast, takes account of prior beliefs about the population $\mathbf{P}$, by basing inferences on a *prior probability distribution* that is combined with a sample $\mathbf{S}$ to produce a *posterior (probability) distribution* $P(\theta|\mathbf{S})$, for the parameter of interest θ. Confidence and prediction intervals are defined within the Bayesian framework.

Let θ be a parameter of the population distribution $\mathbf{P}$ and $\mathbf{S}$ a random sample drawn from $\mathbf{P}$. If θ is viewed as a random variable whose posterior distribution is $P(\theta|\mathbf{S})$, then $[\lambda_L(\mathbf{S}), \lambda_U(\mathbf{S})]$ is a $(1-\alpha)100\%$ *Bayesian confidence interval* for θ if from $P(\theta|\mathbf{S})$ there is a $(1-\alpha)100\%$ probability $\theta \in [\lambda_L(\mathbf{S}), \lambda_U(\mathbf{S})]$. In the Bayesian approach, θ is a random variable and $[\lambda_L(\mathbf{S}), \lambda_U(\mathbf{S})]$ is fixed given availability of $\mathbf{S}$. In the classical approach, it is θ which is fixed and $[\lambda_L(\mathbf{S}), \lambda_U(\mathbf{S})]$ varies with $\mathbf{S}$. If $\mathbf{S}$ is a (univariate) random sample drawn from $\mathbf{P}$ where $\{(x_1, x_2, \ldots, x_n) \in \mathbf{S}, n < p\}$ and $P(x_{n+1}|\mathbf{S})$ is the posterior distribution for x_{n+1}, then $[\lambda_L(\mathbf{S}), \lambda_U(\mathbf{S})]$ is a $(1-\alpha)100\%$ *Bayesian prediction interval* for x_{n+1} if from $P(x_{n+1}|\mathbf{S})$ there is a $(1-\alpha)100\%$ probability $x_{n+1} \in [\lambda_L(\mathbf{S}), \lambda_U(\mathbf{S})]$.

For regression, maximum likelihood based methods estimate single values for each (unknown) parameter of the true regression. The Bayesian approach, in contrast, expresses the uncertainty regarding the true weight vector $\boldsymbol{\Omega}$ as the posterior probability distribution $P(\boldsymbol{\Omega}|\mathbf{S})$ given a sample $\mathbf{S}$. Thus:

$$\begin{aligned} P(\mu_y(\mathbf{x})|\mathbf{S}) &= \int_{\boldsymbol{\Omega}} P(\mu_y(\mathbf{x})|\boldsymbol{\Omega}) P(\boldsymbol{\Omega}|\mathbf{S})\,\mathrm{d}\boldsymbol{\Omega} \\ &\propto \int_{\boldsymbol{\Omega}} P(\mu_y(\mathbf{x})|\boldsymbol{\Omega}) P(\mathbf{S}|\boldsymbol{\Omega}) P(\boldsymbol{\Omega})\,\mathrm{d}\boldsymbol{\Omega}, \end{aligned} \tag{1.32}$$

where $P(\boldsymbol{\Omega})$ is the prior distribution for the weights. It has been shown [17] that with approximations, the latter integral can be solved analytically. If the noise is assumed to be $\varepsilon \sim N(0, \sigma^2)$ and the prior $P(\boldsymbol{\Omega})$ is also assumed to be Gaussian, then a Gaussian posterior distribution $P(\mu_y(\mathbf{x})|\boldsymbol{\Omega}_{\mathrm{MP}})$ can be derived where:

$$\hat{E}[\mu_y(\mathbf{x})] = \hat{\mu}_y(\mathbf{x}; \boldsymbol{\Omega}_{\mathrm{MP}}). \tag{1.33}$$

In (1.33), $\boldsymbol{\Omega}_{\mathrm{MP}}$ is $\boldsymbol{\Omega}$ at the maximum of the posterior probability distribution $P(\boldsymbol{\Omega}|\mathbf{S})$. The variance of $P(\mu_y(\mathbf{x})|\boldsymbol{\Omega}_{\mathrm{MP}})$ is

$$\widehat{\mathrm{Var}}(\mu_y(\mathbf{x})) = (\sigma^2)^{-1} + \mathbf{g}^\top \mathbf{A}^{-1} \mathbf{g}, \tag{1.34}$$

where σ^2 is the (constant) noise variance and $\mathbf{A}^{-1}$ is the Hessian matrix of second-order partial derivatives (with respect to each of the elements of $\boldsymbol{\Omega}$) of the regularised cost function:

$$\frac{\sigma^2}{2} \sum_{i=1}^{N} [\hat{\mu}_y(\mathbf{x}_i; \boldsymbol{\Omega}) - y_i]^2 + \frac{\lambda}{2} \sum_i w_i. \tag{1.35}$$

The second term in (1.35) is a regularisation term resulting from the assumption that $P(\boldsymbol{\Omega})$, the prior distribution in (1.32), is a Gaussian. In (1.35), λ is a constant and the w_i are the weights from (1.11). It follows that an approximate $(1-\alpha)100\%$ Bayesian prediction interval for $\mu_y(\mathbf{x})$ is given by

$$\hat{\mu}_y(\mathbf{x}_0;\boldsymbol{\Omega}_{\mathrm{MP}}) \pm z_{(1-\alpha)100\%}\sqrt{(\sigma^2)^{-1} + \mathbf{g}_0{}^{\top}\mathbf{A}^{-1}\mathbf{g}_0}. \tag{1.36}$$

By using (1.36), maximum likelihood estimation has been avoided. However, the derivation relies on the same assumptions of normality of the errors and unbiasedness as the delta method, to which it is related. The method has been extended by Bishop and Qazaz [4] to the case of non-constant variance, by replacing the constant noise variance term in (1.34) with input-dependent (variable) noise variance. An advantage of the Bayesian approach is that the regularisation parameter is automatically determined during training. This means cross validation is not required to control over-fitting, so all of the available data can be used for training. Unfortunately, obtaining Bayesian standard error estimates is substantially more complex than using maximum likelihood based approaches [25]. This is due to the need to use approximations to obtain the analytical formulae. The Bayesian method is unreliable where crude approximations are used. Moreover, inversion of the Hessian matrix is required, with the attendant possibility of failure.

1.3 Implementations of the Theory

In this section, the empirical performance of different implementations of the theory outlined in Sect. 1.2 for estimating standard errors, confidence, and prediction intervals for computational intelligence techniques used for regression is considered.

Tests of eight different standard error estimates for MLPs with single hidden layers and a linear output layer were performed by Tibshirani [24]. Four variants of the delta method, two variants of the sandwich estimator, the bootstrap pairs and bootstrap residuals methods were tested. The delta method variants tested by Tibshirani [24] were the standard, a method using the inverse Hessian matrix, a method using an approximation to the Hessian matrix omitting second-order terms, and the delta method with a regularisation term. The sandwich method variants were the standard sandwich method and a variant using an approximate Hessian matrix. A small data set was used for these tests, consisting of 111 observations on air pollution. For the bootstrap methods, $B = 20$ bootstrap replicates were used. The bootstrap methods provided the most accurate estimates of standard errors. The delta methods and sandwich estimators missed the substantial variability due to random starting weights. Tibshirani [24] suggests these latter estimators may perform better where there is less sensitivity to the choice of starting weights (initialisation parameters), for example with larger data sets where gradient descent is used.

This suggestion is consistent with findings reported by LeBaron and Weigend [16] who used a training set of 3,200 observations on market trading volume at the New York Stock Exchange. These data were relatively noisy, with predictions explaining approximately 0.5 of the variance. Both an MLP and a linear model were tested. The error measure used by LeBaron and Weigend [16] was $(1-R^2)$, and 2,523 bootstrap replicates were generated on the test set of 1,500 observations to obtain out-of-sample distributions for this error. For the MLP, 697 networks were also trained on a single sample and initialisation parameters were randomly drawn for each one. It was found that the randomness due to the splitting of the data generated more variability than the randomness due to network initialisation. No significant correlation was found between the choice of initialisation parameters or network topology and performance. Moreover, the error distributions obtained from the bootstrap procedure on the test set were almost identical for both the linear and MLP models.

In tests using synthetic data with an input-dependent noise variance, Bishop and Qazaz [4] demonstrated that the Bayesian approach can give an improved estimate of noise variance compared with maximum likelihood based approaches. However, Ungar et al. [25] question whether the improved performance of Bayesian approaches justifies the extra computational cost involved.

The method proposed by Heskes [10] uses bootstrap pairs to obtain prediction intervals for a pair $\{(\mathbf{x}_0, y_0) \notin \mathbf{S}_b^{\text{Boot}}, b = 1, \ldots, B\}$ using the relationship in (1.10), (1.26) and (1.27). To achieve this a separate neural net $\chi^2(\mathbf{X})$ is trained to model the noise variance $\text{Var}(\varepsilon)$. The targets for this network are residuals satisfying

$$r^2(\mathbf{X}_v) = \max\left(e_{\text{Boot}}^2(\mathbf{X}_v) - \widehat{\text{SE}}_{\text{Boot}}^2(f(\mathbf{X}_v; \boldsymbol{\Omega})), 0\right) \tag{1.37}$$

obtained from the validation sets used for training the B networks used in the bootstrap ensemble (1.27). Alternatively, these may be obtained by applying (1.27) to an independent test set. In (1.37), $e_{\text{Boot}}^2(\mathbf{X}_v) = (y - \hat{\mu}_{y,\text{Boot}(\mathbf{X}_v)})^2$, the residuals from (1.27) when applied to the validation sets. The cost function used for training the auxiliary network is the negative log likelihood function; hence the use of the $\max(\cdot, 0)$ function in (1.37). The resulting bootstrap prediction interval is

$$\hat{\mu}_{y,\text{Boot}}(\mathbf{X}) \pm t_{(\alpha/2)B}\left(\widehat{\text{SE}}_{\text{Boot}}(f(\mathbf{X}; \boldsymbol{\Omega})) + \chi(\mathbf{X})\right). \tag{1.38}$$

The prediction interval (1.38) offers the advantage that it allows for the uncertainty of the regression function as well as that of the noise. In addition, it does not rely on the assumption that the network is an unbiased estimator of the conditional mean of the target value (i.e. that the error due to bias is negligible compared with the error due to variance).

A novel method of computing prediction intervals for neural nets has been proposed by Nix and Weigend [19]. The method uses an NN with two output nodes, one for $\hat{\mu}_y(\mathbf{x}; \hat{\boldsymbol{\Omega}})$, the predicted value and a second for $\hat{\sigma}_y^2(\mathbf{x}; \hat{\mathbf{U}})$, the variance of the predicted value. The network has a non-standard structure, with a second hidden layer for $\hat{\sigma}_y^2(\mathbf{x}; \hat{\mathbf{U}})$ receiving inputs from both the hidden layer for $\hat{\mu}_y(\mathbf{x}; \hat{\boldsymbol{\Omega}})$ and from the input layer. A negative log likelihood cost function is used, modified by inclusion of the input-dependent variance term $\hat{\sigma}_y^2(\mathbf{x}_i)$. Usually $\hat{\sigma}^2(\mathbf{X})$ is assumed constant and drops out after differentiation. A linear activation function is specified for the $\hat{\mu}_y(\mathbf{x};\hat{\boldsymbol{\Omega}})$ output unit. To ensure only positive outputs, an exponential activation function is specified for the $\hat{\sigma}_y^2(\mathbf{x};\hat{\mathbf{U}})$ output unit. A hyperbolic tangent activation function is used for the hidden layer units. Using these activation functions, differentiating the cost function with respect to the network weights gives weight update equations containing terms $1/\hat{\sigma}_y^2(\mathbf{x})$, which act as a form of weighted regression. An improved fit in low noise regions of the input space is claimed by Nix and Weigend [19]. The outputs obtained from this network are equivalent to training a separate network for $\hat{\sigma}_y^2(\mathbf{x};\hat{\mathbf{U}})$ using the squared residuals from $\hat{\mu}_y(\mathbf{x};\hat{\boldsymbol{\Omega}})$ as targets.

The network proposed by Nix and Weigend [19] requires a three-phase training process. In *Phase I* the network is trained on a training set $\mathbf{A}$ for the output $\hat{\mu}_y(\mathbf{x};\hat{\boldsymbol{\Omega}})$. This is equivalent to normal network training using a sum of squares cost function with early stopping to prevent over-fitting. In *Phase II*, the weights trained in Phase I are frozen and the second hidden layer for $\hat{\sigma}_y^2(\mathbf{x};\hat{\mathbf{U}})$ added. The squared residuals from the Phase I model are used as targets for the second output node $\hat{\sigma}_y^2(\mathbf{x};\hat{\mathbf{U}})$. The network is now trained for the output $\hat{\sigma}_y^2(\mathbf{x};\hat{\mathbf{U}})$ using the validation set $\mathbf{B}$ from Phase I as the training set, with set $\mathbf{A}$ as the validation set. In Phase III (weighted regression), the available data are re-split into a new training set $\mathbf{A}'$ and validation set $\mathbf{B}'$. All network weights are unfrozen, and the network is re-trained for both output nodes on training set $\mathbf{A}'$, using $\mathbf{B}'$ as the validation set. Training is now considered complete.

The method of Nix and Weigend [19] can provide prediction intervals without bootstrap re-sampling or use of the Hessian matrix. Improved performance in low noise regions of the input space is claimed, due to the form of weighted regression used. However, the use of weighted regression means the standard errors obtained must be interpreted with caution. This is because inference for NNs and other computational intelligence techniques rests on the assumption that an NLLS regression is performed, as discussed in Sect. 1.2.2. In weighted regression a penalty term is added to the least squares cost function and this may not be the case. Moreover, as Nix and Weigend [19] themselves point out, weighted regression introduces local minima in the error surface, complicating learning. Improved prediction intervals are claimed by Nix and Weigend [19] in tests using both synthetic data with added non-uniform Gaussian noise and real-world data with uniform non-Gaussian noise, compared to the use of a separate network to estimate the variance, as proposed by Satchwell [20] or used by Heskes [10].

1.3.1 Limitations of Existing Approaches

Existing methods for computing standard errors, confidence, and prediction intervals for computational intelligence techniques used for regression are of three types: first, the delta method, sandwich method, and their variants, which use the Hessian matrix of second-order partial derivatives of the cost function with respect to the weights and biases; second, methods using bootstrap re-sampling; third, methods which rely on directly modelling the noise.

The empirical findings considered above suggest that methods of the first type do not perform as well as methods of the second type, at least for small samples. Moreover, where there is over-fitting the matrix inversions required for the Delta and Sandwich methods are unstable and may fail, making computations impossible. Methods based on bootstrap re-sampling are reported by Tibshirani [24] to give estimates that are more accurate. While the bootstrap can provide confidence intervals for the true regression function, on its own it cannot provide prediction intervals where the target variable is unknown. Heskes [10] attempts to overcome this limitation by employing a separate neural network to model the noise, in conjunction with bootstrap re-sampling. However, the naïve bootstrap does not provide heteroskedasticity consistent standard errors [26]. For these, the use of a more complex wild bootstrap is required. The method proposed by Nix and Weigend [19] is of the third type. It does not have any of the above limitations, and is less computationally costly. However, it relies on a non-standard architecture, requiring special programming, a complex training algorithm, and utilises a form of weighted regression.

1.4 Robust Practical Prediction Intervals

A robust practical method for obtaining standard errors, confidence, and prediction intervals that is applicable to any computational intelligence technique of sufficient generality used for regression is now described [9]. The method is robust to heteroskedasticity and practical to implement. It allows the standard error to be obtained directly and avoids the bootstrapping that is otherwise a practical necessity in obtaining confidence intervals for the true regression.

The method is based on a network with two outputs, one fitted to the target variable and the other to its squared error. It differs from the method of Nix and Weigend [19] because: (a) It uses a sum of squares cost function and does not assume a Gaussian noise distribution. (b) It uses a training algorithm with independent training and validation sets, rather than interchanging validation sets. This latter feature is due to a suggestion by Heskes [10] that it is desirable that the training set for fitting the squared errors is disjoint from either the training or validation sets used for fitting the target variable. The reason is, when a network is trained using early stopping, training is stopped when the sum of squared errors is minimised not on the training set, but on a separate validation set. Thus, the model obtained is a function of both data sets, and the validation set cannot be viewed as independent for the purpose of fitting the squared errors. The theoretical basis of the method follows.

1.4.1 Least Squares Derivation

The object in training (fitting) an MLP or other computational intelligence technique is not to memorise features specific to the training set, but to model the underlying data generating process, so that when presented with a new input vector $\mathbf{x}$, the trained network gives the best possible estimate of the target. The most comprehensive description of the DGP is a statistical one, in terms of the joint probability density $P(\mathbf{x},\mathbf{d})$ of the input vector $\mathbf{x}$ and the target vector $\mathbf{d}$. This density can be expressed as the product of the conditional distribution $P(\mathbf{d}|\mathbf{x})$ of the target vector $\mathbf{d}$ conditioned on the input vector $\mathbf{x}$ and the unconditional distribution $P(\mathbf{x})$ of the input vector

$$P(\mathbf{x},\mathbf{d}) = P(\mathbf{d}|\mathbf{x})P(\mathbf{x}), \tag{1.39}$$

where

$$P(\mathbf{x}) = \int P(\mathbf{x},\mathbf{d})\,\mathrm{d}\mathbf{d}. \tag{1.40}$$

An MLP trained by minimising a sum of squares error defined over a training set approximates the means of the elements of a target vector $\mathbf{d}$ conditioned on a corresponding input vector $\mathbf{x}$. The optimisation results in estimation of a vector $\hat{\boldsymbol{\Omega}}$ of weights and biases that minimise the cost function. The function to be minimised takes the form

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}[f_j(\mathbf{x}_i;\hat{\boldsymbol{\Omega}}) - d_{ij}]^2. \tag{1.41}$$

In (1.41) d_{ij} is the jth element of the ith target vector. $f_j(\mathbf{x}_i;\hat{\boldsymbol{\Omega}})$ is the corresponding network estimate. Asymptotically as n, the size of the data set, tends to infinity and assuming the function $f_j(\mathbf{x}_i;\hat{\boldsymbol{\Omega}})$ has sufficient flexibility (i.e. degrees of freedom), bias and variance tend to zero yielding the optimum least squares solution.

In the limit, the summation over n in (1.41) becomes an integral over the joint probability density [3]:

$$C^{\mathrm{LS}} = \lim_{n\to\infty}\frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{m}[f_j(\mathbf{x}_i,\hat{\boldsymbol{\Omega}}) - d_{ij}]^2 \tag{1.42}$$

$$= \frac{1}{2}\sum_{j=1}^{m}\iint[f_j(\mathbf{x},\hat{\boldsymbol{\Omega}} - d_j]^2 p(d_j|\mathbf{x})P(\mathbf{x})\,\mathrm{d}d_j\,\mathrm{d}\mathbf{x}, \tag{1.43}$$

where $1/n$ in (1.43) is a convergence factor. The cost function can be minimised (using functional differentiation) with respect to $f(\mathbf{x},\hat{\boldsymbol{\Omega}})$ and setting the derivative to zero

$$\frac{\delta C^{\mathrm{LS}}}{\delta f_j(\mathbf{x},\hat{\boldsymbol{\Omega}})} = 0. \tag{1.44}$$

Substituting (1.43) into (1.44) and using (1.39) yields the following solution for the minimising function:

$$f_j(\mathbf{x}, \hat{\boldsymbol{\Omega}}) = E(d_j|\mathbf{x}) = d_j(\mathbf{x}). \tag{1.45}$$

Thus, the output of the network function corresponds to the conditional means of the elements of the target vector **d**, conditioned on the input vector **x**. *The result (1.45) depends only on the generality of the non-linear mapping represented by the network function. It does not specifically require use of an MLP and thus extends to any comparable non-linear mapping of sufficient flexibility*.

The (global) conditional variance corresponding to the conditional mean (1.45) is given by

$$\mathrm{Var}(d_j|\mathbf{X}) = \frac{\sum_{i=1}^{n} e_{ij}^2}{(n-k-1)}, \tag{1.46}$$

where $\mathbf{X}$ is the matrix of input data, e_{ij}^2 is the squared residuals for $d_j(\mathbf{x}_i)$ at the minimum of the cost function, n is the number of observations in the data set, and k is the applicable degrees of freedom. Given (1.46) the conditional distribution of the target $P(d_j|\mathbf{x}_i)$ is characterised by a two parameter distribution with a mean given by $f_j(\mathbf{x}, \hat{\boldsymbol{\Omega}})$ and a (global) variance given by $\mathrm{Var}(d_j|\mathbf{X})$. However, the use of a least squares cost function does *not* require the assumption that this distribution is Gaussian.

Suppose that the target vector **d** as well as d_j have an additional element $\sigma_j^2(\mathbf{x})$ where $\sigma_j^2(\mathbf{x}) = \{d_j - E(d_j|\mathbf{x})\}^2$, the squared residuals from the network estimate of $d_j(\mathbf{x})$. Then it follows from (1.45) that:

$$\sigma_j^2(\mathbf{x}) = E[\{d_j - E(d_j|\mathbf{x})\}^2|\mathbf{x}] = \mathrm{Var}(d_j|\mathbf{x}). \tag{1.47}$$

The function $\hat{\sigma}_j^2(\mathbf{x})$ is modelled by adding an additional output node to the MLP trained to fit the squared residuals of $\hat{d}_j(\mathbf{x})$. Using (1.47) in place of (1.46) allows estimation of a separate variance parameter for each target d_j conditioned on the corresponding input vector **x**, and is equivalent to using White's heteroskedasticity consistent estimator [23].

1.4.2 Maximum Likelihood Derivation

If the conditional distribution of the target data is assumed to be Gaussian, the result (1.45) can be derived using maximum likelihood [2]. Under the Gaussian assumption $P(d_j|\mathbf{x})$ can be written as

$$P(d_j|\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{[f_j(\mathbf{x};\hat{\boldsymbol{\Omega}}) - d_j]^2}{2\sigma^2}\right), \tag{1.48}$$

where σ^2 is a global variance parameter that can be estimated by (1.46). Again, this is easily extended to obtain a more general distributional assumption by substituting (1.47) for (1.46) in (1.48) giving

$$P(d_j|\mathbf{x}) = \frac{1}{(2\pi)^{1/2}\sigma_j(\mathbf{x})}\exp\left(-\frac{[f_j(\mathbf{x};\hat{\boldsymbol{\Omega}}) - d_j]^2}{2\sigma_j^2(\mathbf{x})}\right). \tag{1.49}$$

Maximising the likelihood is equivalent to minimising the negative log likelihood. Forming the negative log likelihood of (1.49) and omitting constants give

$$C^{-\mathrm{LL}} = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(\ln\sigma_j(\mathbf{x}_i) + \frac{[f_j(\mathbf{x}_i,\hat{\boldsymbol{\Omega}}) - d_{ij}]^2}{2\sigma_j^2(\mathbf{x}_i)}\right). \tag{1.50}$$

Taking the limit as before gives the integral form

$$C^{-\mathrm{LL}} = \sum_{j=1}^{m}\iint\left(\ln\sigma_j(\mathbf{x}) + \frac{[f_j(\mathbf{x},\hat{\boldsymbol{\Omega}}) - d_j]^2}{2\sigma_j^2(\mathbf{x})}\right)P(d_j|\mathbf{X})P(\mathbf{x})\,\mathrm{d}d_j\,\mathrm{d}\mathbf{x}. \tag{1.51}$$

Functional differentiation is again used to minimise the errors for the network outputs for the mean and variance functions. For the mean:

$$\frac{\delta C^{-\mathrm{LL}}}{\delta f_j(\mathbf{x},\hat{\boldsymbol{\Omega}})} = 0 = P(\mathbf{x})\int\frac{[f_j(\mathbf{x},\hat{\boldsymbol{\Omega}}) - d_j]}{\sigma_j^2(\mathbf{x})}P(d_j|\mathbf{x})\,\mathrm{d}d_j. \tag{1.52}$$

Rearranging and simplifying (1.52) give the standard result of (1.45). For the variance, (1.51) is minimised with respect to the function $\sigma_j(\mathbf{x})$ giving

$$\frac{\delta C^{-\mathrm{LL}}}{\delta\sigma_j(\mathbf{x})} = 0 = P(\mathbf{x})\int\left(\frac{1}{\sigma_j(\mathbf{x})} - \frac{[f_j(\mathbf{x},\hat{\boldsymbol{\Omega}}) - d_j]^2}{\sigma_j^2(\mathbf{x})^3}\right)P(d_j|\mathbf{x})\,\mathrm{d}d_j. \tag{1.53}$$

Using (1.45) again and solving for $\sigma_j^2(\mathbf{x})$ give (1.47). This approach is based on maximum likelihood and gives a biased estimate of the variance, because it makes use of an estimated mean, rather than the (unknown) true mean. The relationship between the true variance and its estimate obtained under maximum likelihood is given by

$$\hat{\sigma}^2 = \frac{n-k-1}{n}\sigma^2, \text{ hence } \sigma^2 = \frac{n}{n-k-1}\hat{\sigma}^2, \tag{1.54}$$

where k is the appropriate degrees of freedom [3].[1]

[1]Equation (1.54) also applies to least squares estimators. The value of k depends on the particular regression technique used.

Table 1.1 Training algorithm

Phase	Description
I	(a) Randomly split the training data into two data sets, set A and set B
	(b) Using set A, train a Phase I NN to fit the target variable $d(\mathbf{x})$
	(c) Run the trained NN model on set B, create a set of squared residuals
II	(a) Using set B, train a Phase II NN with two output nodes Use the variable $d(\mathbf{x})$ as the target for the first output node Use the squared residuals created in Phase I using set B as the target for the second output node
III (optional)	(a) Run the Phase II model on set A, create a set of squared residuals for the target $d(\mathbf{x})$
	(b) Using set A, train a Phase III NN with two output nodes Use the variable $d(\mathbf{x})$ as the target for the first output node Use the squared residuals created in step III (a) using set A as the target for the second output node
Notes	(1) By using squared residuals on a test set (set B) as the second target for Phase II, over-fitting and consequent underestimation of the standard error is avoided
	(2) Phase III may give improved results on certain data, but generally produces inferior results to Phase II, and may thus be omitted. For training, set A is itself randomly split into a training and a validation portion; as is set B. Testing of each phase is performed on an independent test set, set C

Thus, it is shown here that an MLP with two output nodes, the first trained to fit a target value, and the second trained to fit the squared residuals of the first fit, can produce an estimate of the mean and variance of the conditional distribution of the target in both the least squares and maximum likelihood frameworks. The maximum likelihood derivation requires the assumption that the conditional distribution of the target data is Gaussian. This assumption is not made for the proposed method. However, maximum likelihood and least squares estimators are otherwise equivalent. *Moreover, the result (1.45) which is central to the proposed method requires only the use of a least squares cost function, and a sufficiently flexible form of non-linear regression, thus encompassing a wide variety of computational intelligence techniques used for regression.*

1.4.3 Practical Implementation and Performance

Practical application of the proposed method requires the use of a special training algorithm. The theory presented in Sects. 1.4.1 and 1.4.2 is implemented using the algorithm given in Table 1.1.

1.4.3.1 Empirical Tests and Performance

The method has been extensively tested in [9] using the following sequence. First the method was tested as Example #1, using the same synthetic data and single input variable used by Nix and Weigend [19]. This allowed a performance comparison with their method. Next the method was tested as Example #2, using option market data. Synthetic option prices and noise generated by known underlying functions were used. Example #2 is deliberately restricted to two input variables, moneyness and maturity $[m,t]$, following [14]. This allowed the use of a known smooth noise variance function. The purpose of Example #2 was to test the method in a more realistic multi-dimensional, option pricing context, while retaining comparability with Example #1. The method was next tested using actual option prices, as Example #3. The same option market data as used for Example #2 were again used. For comparability with Example #2 the model was again restricted to the two inputs [m,t]. The underlying true regression function and noise variance function were unknown for Example #3, however, in contrast to Examples #1 and #2 where synthetic prices and noise were used, generated by known functions. Finally, the method was tested as Example #4 using actual prices and all five standard inputs to the Black–Scholes formula. The same data sets were again used.

The sequence of examples features increasing dimensionality from the one-dimensional Example #1, through the two-dimensional Examples #2 and #3, to the five-dimensional Example #4. The sequence also relaxes the condition of synthetic inputs, target data, and noise, used for Example #1. It moves through the synthetic target and noise to the actual prices, market inputs, and residuals of Examples #3 and #4. At the beginning of the sequence, a direct comparison was possible with the method of Nix and Weigend [19], and at the end, a direct comparison with the benchmark BS formula was possible.

1.4.3.2 Example #1: Tests Using Standard Synthetic Data

Nix and Weigend [19] defined a univariate synthetic example to demonstrate the effectiveness of their model. For comparison purposes the proposed training algorithm and network were applied to the same univariate synthetic data and called Example #1 in their paper and here. This test example used a one-dimensional data set where $y(\mathbf{x})$, the true regression, and $\sigma^2(\mathbf{x})$, the variance of the noise, were known. The true regression $y(\mathbf{x})$ is given by the equation $y(\mathbf{x}) = \sin(3x)\sin(5x)$, where x is a uniformly distributed random number from the interval $[0, \pi/2]$. The noise $n(\mathbf{x})$ consists of numbers from the normal distribution $N[0, \sigma^2(\mathbf{x})]$, where $\sigma^2(\mathbf{x}) = 0.02 + 0.25[1 - \mathrm{Sin}(5x)]^2$.[2] The target value for training is $d(\mathbf{x}) = y(\mathbf{x}) + n(\mathbf{x})$.

[2]Heskes [10] used similar trigonometric functions for the true regression and the noise variance.

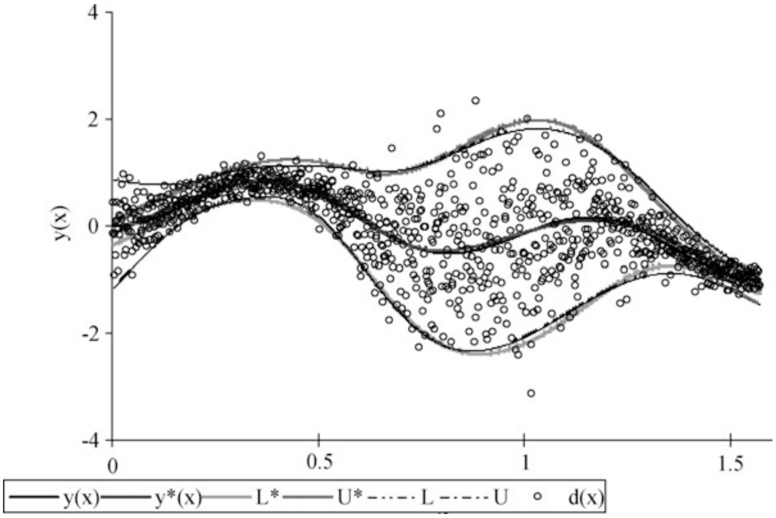


Fig. 1.1 Prediction bands for Example #1. Here, $y(\mathbf{x})$ is the true regression. $d(\mathbf{x})$ are the target data points. $y^*(\mathbf{x})$ is the estimate of the true regression. L and U are the true lower and upper prediction intervals and L^* and U^* are the estimated prediction intervals obtained using $\sigma^{*2}(\mathbf{x})$, the network estimate of the noise variance function

The following procedure was adopted. For Phase I, a network with a single input node, 10 hidden layer nodes and a single output node to fit $d(\mathbf{x})$, was used. Phases II and III used a network with a single input node, 20 hidden layer nodes and 2 output nodes, one to fit $d(\mathbf{x})$ and one to fit $\sigma^2(\mathbf{x})$. The numbers of hidden nodes used were the same as in [19], except that here they were fully connected in a single layer. Figure 1.1 shows a plot of the data points, the true regression $y(\mathbf{x})$, and the approximate prediction band, for the Phase III model, obtained on a test set. The effect of Phase III was to improve the Adj. R^2 figure and the F-statistic, for the estimated noise variance function. Table 1.2 shows the results of statistical tests for Example #1 in predicting the true regression, the true noise variance function, the target data points $d(\mathbf{x})$, and the actual squared residuals.

In Table 1.2 $\mu_y^*(x)$ is the estimate produced by the node having $d(\mathbf{x})$ as target, and $\sigma^{*2}(\mathbf{x})$ is the estimate produced by the node having squared residuals as target. In the comparison of the network estimates with the true regression function and noise variance function, unbiased estimates of the mean of the true regression $y(\mathbf{x})$ and the true noise variance function $\sigma^2(\mathbf{x})$ were obtained for both Phases II and III. For Phase III, the F-statistics suggest the distributions of values for the true regression and true noise variance functions were also well recovered. Approximate upper and lower prediction intervals based on the estimated noise variance function were also unbiased estimates of the true upper and lower prediction intervals. The fit to the true regression function is very good in Phases II and III with R^2 and Adj. R^2 figures > 0.99. The fit to the noise variance function was also excellent with Phases II and III R^2 and Adj. R^2 figures > 0.94 in all cases. In the comparisons of

Table 1.2 Estimates by proposed network (Example #1)

	Layers	Inputs				F-stat		t-statistic		t-test
Phase	[Nodes]	x	Outputs	R^2	Adj. R^2	F_{crit}(1 tail)	$F_{\text{calc}}(0.05)$	t_{crit}(2 tail)	$t_{\text{calc}}(0.05)$	[Bias]
Comparison with true regression function										
I	1-10-1	x	$\mu_y^*(x)$	0.561	0.561	1.03	1.75	1.96	0.28	Unbiased
Comparison with true regression function and noise variance function										
II	1-20-2	x	$\mu_y^*(x)$	0.994	0.994	1.03	1.01	1.96	0.95	Unbiased
II	1-20-2	x	$\sigma^{*2}(x)$	0.965	0.944	1.03	1.74	1.96	1.64	Unbiased
III	1-20-2	x	$\mu_y^*(x)$	0.992	0.991	1.03	1.01	1.96	−1.12	Unbiased
III	1-20-2	x	$\sigma^{*2}(x)$	0.987	0.989	0.97	0.86	1.96	−1.96	Unbiased
Comparison with actual target (d)										
I	1-10-1	x	$\mu_y^*(x)$	0.234	0.234	1.03	4.18	1.96	0.15	Unbiased
Comparison with actual target (d) and squared residuals										
II	1-20-2	x	$\mu_y^*(x)$	0.423	0.423	1.03	2.41	1.96	0.68	Unbiased
II	1-20-2	x	$\sigma^{*2}(x)$	0.211	0.967	1.03	4.41	1.96	−8.55	Biased
III	1-20-2	x	$\mu_y^*(x)$	0.422	0.422	1.03	2.4	1.96	−0.91	Unbiased
III	1-20-2	x	$\sigma^{*2}(x)$	0.338	0.337	1.03	4.38	1.96	0.32	Unbiased

The proposed network produces unbiased estimates $\mu_y^*(x)$ of the underlying true regression function, and $\sigma^{*2}(x)$ of the noise variance function. Phase III estimates of the actual target d and actual squared residuals are also unbiased. The t-test for the means shows no difference at the 95% level

Table 1.3 Results for Example #1: methods compared

		This work		Nix–Weigend	
		Test set	$(n = 10^4)$	Test set	$(n = 10^5)$
Row	Target d	E_{NMS}	Our mean cost	E_{NMS}	NW mean cost
1	Phase I	0.764	0.454	0.593	0.882
2	Phase II	0.577	0.344	0.593	0.566
3	Phase III	0.578	0.344	0.57	0.462
4	*$n(x)$ (exact additive noise)*	*0.575*	*0.343*	*0.563*	*0.441*
	Target e^2	ρ (PIII)	ρ (PII)	ρ (PIII)	ρ (PII)
5	$\rho(\sigma^*(x)$, residual errors)	0.571	0.569	0.548	N/a
6	$\rho(\sigma(x)$, *residual errors*)	*0.586*	*0.585*	*0.584*	N/a
	Distribution P(III)	1 std.	2 std.	1 std.	2 std.
7	% of errors $< \sigma^*(x)$; $2\sigma^*(x)$	67.4	93.1	67.0	94.6
8	*% of errors* $< \sigma(x)$; $2\sigma(x)$	*66.9*	*95.0*	*68.4*	*95.4*
9	(*exact Gaussian*)	*68.3*	*95.4*	*68.3*	*95.4*

E_{NMS} is the mean squared error normalised by the (global) variance of the target d. The mean cost is the mean of the cost function $(d - d^*)^2$. Row 4 gives these figures for $[(d - y(x))^2 = n(x)^2]$ and represents the best performance attainable. Row 5 gives the correlations between the absolute errors and the network estimate for the standard deviation of the errors. Row 6 gives the correlations between the absolute errors and the true noise standard deviation. Row 7 gives the percentage of absolute errors that are less than 1 and 2 times the corresponding network estimate for the standard deviation of the error. Row 8 gives the percentage of absolute errors that are less than 1 and 2 times the corresponding true noise standard deviation. Row 9, which is included for comparison purposes, gives the percentage of observations that are less than 1 and 2 standard deviations in a Gaussian distribution

the network estimates with the actual target $d(\mathbf{x})$ and the actual squared residuals, the much poorer R^2 , Adj. R^2, and F-statistic figures are consistent with the noisy, scattered, target data points. However, in both Phases II and III, the t-test results indicate unbiased estimates of the mean of the target d. In Phase III, the estimate of the mean of the squared residuals is also unbiased.

For comparison of the proposed network performance, the statistics used by Nix and Weigend [19] were computed. These are shown in Table 1.3 for Example #1 and Table 1.4 for Example #2. Table 1.4 results are discussed in Sect. 1.4.3.3. Table 1.3 shows that compared to the network of Nix and Weigend [19] there was little improvement in the fit to the target $d(\mathbf{x})$ between Phases II and III. However, the Phase III fit (row 3) for the proposed network was close to the best attainable, deviating only by 0.6%. The method of Nix and Weigend [19] does not approach the best attainable figure quite so closely, deviating by 1.4%. The proposed network figures for correlation of the actual absolute errors with the network prediction and the true values (rows 5 and 6) improved slightly on the corresponding figures for the Nix–Weigend network, even in Phase II. The distributions of errors reported in rows 7 and 8 differed only slightly from those of Nix and Weigend [19]. Overall, the results in Table 1.3 show the proposed network performed comparably with a Nix–Weigend network [19] and outperformed it slightly on the errors and correlations. However, the results in Table 1.3 are not based on hypothesis tests or confidence

Table 1.4 Results for Example #2: synthetic option prices + noise

		Test set	
Row	Target $d = C_{\mathrm{NN}} + \text{noise}$	E_{NMS}	Our mean cost
1	Phase I	0.059	564.6
2	Phase II	0.062	599.4
3	Phase III	0.059	566.8
4	$n(\mathbf{x})$ (*exact additive noise*)	*0.058*	*556.9*
	Target e^2	ρ (PIII)	ρ (PII)
5	$\rho(\sigma^*(\mathbf{x})$, residual errors)	0.537	0.536
6	$\rho(\sigma(\mathbf{x})$, residual errors)	0.562	0.591
	Distribution (PII)	1 std.	2 std.
7	% of errors $< \sigma^*(\mathbf{x}); 2\sigma^*(\mathbf{x})$	51.30	81.40
8	% of errors $< \sigma(\mathbf{x}); 2\sigma(\mathbf{x})$	70.00	96.10
	Distribution (PIII)		
9	% of errors $< \sigma^*(\mathbf{x}); 2\sigma^*(\mathbf{x})$	32.50	60.20
10	*% of errors* $< \sigma(\mathbf{x}); 2\sigma(\mathbf{x})$	*71.40*	*96.20*
11	(*exact Gaussian*) (%)	*68.30*	*95.40*

Rows 1–8 are as Table 1.3. Rows 9 and 10 for Phase III correspond to rows 7 and 8 for Phase II. Row 11 corresponds to row 9 in Table 1.3. Here ($\mathbf{x}$) represents the vector of input variables $[t, m]$

intervals; therefore the results presented in Table 1.2 are preferred. The results presented in Table 1.2 showed that the proposed network was able to provide unbiased estimates of an underlying true regression function, an associated noise variance function, and the actual targets and squared errors in the univariate case.

1.4.3.3 Example #2: Tests Using Synthetic Option Prices

This section reports results for Example #2 where the proposed network was applied in the more realistic multivariate setting of option pricing. The method was tested using synthetic option prices and synthetic noise. The option market data used were from LIFFE. They consisted of daily closing prices for the FTSE-100 index call option for all trading dates from 13 March 1992 to 1 April 1997. The raw data set contained 119,413 records. The data were cleaned to remove illiquid contracts. The cleaned data set comprised 14,254 records. This data set was randomly split into a training set and a test set. The resulting training sets contained 7,083 records with 3,629 in Set A and 3,454 in Set B. Fifty percent of these were randomly sampled and used for validation. The test set contained 7,171 records.

The synthetic option prices were created using a trained neural net option pricing model as the underlying known true regression function. For this purpose, the approach of Hutchinson et al. [14] was followed, and the volatility and risk-free interest rate were omitted as inputs. The network was trained using observed market prices as the target and the variables moneyness, ($m = S/X$), and time to maturity, t, as inputs; S represents the price of the asset in index points and X is the strike price for the option. Analysis of squared residual errors for neural net option pricing models indicated that $\sigma^2(t,m) = 510t^4 + 361m^{17}$ was an acceptable parameterisation for the known true noise variance function for approximating the underlying residuals; it is important to emphasise that this function has no significance other than providing a noise variance model for this example. Using $\sigma^2(t,m)$, a synthetic noise distribution very similar to that for real residual errors for neural net option pricing models was obtained, as indicated by variance, standard deviation, skewness, and kurtosis. Synthetic noise from the normal distribution N was drawn as $N(0,\sigma^2(t,m))$ and added to the outputs of the trained NN to generate a synthetic target option price $d(t,m)$. The obtained target $d(t,m)$ was not significantly different from observed market prices in t-tests and F-tests. As in Example #1, the aim was to determine whether the method could successfully recover an underlying known regression function and noise variance function. The results for Example #2 are presented in Table 1.5.

In Table 1.5 $\mu_y^*(\mathbf{x})$ is the estimate produced by the node having $d(\mathbf{x})$ as target, and $\sigma^{*2}(\mathbf{x})$ is the estimate produced by the node having the squared residuals as target. In the comparison of the network estimates with the true regression function and noise variance function, unbiased estimates of the mean of the true regression $y(\mathbf{x})$ and the true noise variance function $\sigma^2(\mathbf{x})$ were obtained for Phase II. The Phase III estimate of the mean of the noise variance function $\sigma^2(\mathbf{x})$ was biased. For both Phases II and III, the F-statistics suggest the distribution shapes for the true noise variance function were again well recovered. The fits to both the true regression and noise variance function for both Phases II and III are again very good, as measured by R^2 and Adj. R^2. In the comparisons of Example #2 estimates with the actual target $d(\mathbf{x})$ and the actual squared residuals, the R^2, and Adj. R^2 figures for $\mu_y^*(\mathbf{x})$ are much better than the corresponding results for Example #1. This is because the synthetic option price data are far less noisy than the corresponding data used for Example #1. Again, in both Phases II and III, the t-test results indicate unbiased estimates of the mean of the target $d(t,m)$. In Phase II, the estimate of the mean of the squared residuals is also unbiased.

For comparison of the proposed network performance with that of Nix and Weigend [19], the statistics used by Nix and Weigend [19] were again computed. These are shown in Table 1.4 in Sect. 1.4.3.2. In Table 1.4 the Phase III fit for the target $d(t,m)$ (Row 3) is only 0.001, (1.7%) more than the lowest attainable value (Row 4). The Phase II fit (Row 2) was not quite as good but still only 0.004 (7%) greater than the lowest attainable value. Although there was no statistically significant difference at the 95% level in the t-tests, the Phase II fit to E_{NMS} was slightly poorer than the Phase I fit (Row 1). This may be because, in contrast to [19], the Phase II used here involved training a new model constrained to fit both the target $d(t,m)$ and the squared residuals from Phase I. As with [19] 10 hidden layer nodes per output node were used, but these were arranged in a single hidden layer of 20 nodes with full connectivity to all input and output nodes. Pruning runs, not reported here, indicated fewer nodes could achieve the relevant accuracy.

Table 1.5 Estimates by proposed network (Example #2)

	Layers	Input				F-stat		t-statistic		t-test
Phase	[Nodes]	$\mathbf{x} = [t, m]$	Outputs	R^2	Adj. R^2	F_{crit}(1 tail)	F_{calc}(0.05)	t_{crit}(2 tail)	t_{calc}(0.05)	[Bias]
Comparison with true regression function										
I	2-10-1	$\mathbf{x}$	$\mu_y^*(\mathbf{x})$	1.000	0.998	1.04	1.08	1.96	−0.02	Unbiased
Comparison with true regression function and noise variance function										
II	2-20-2	$\mathbf{x}$	$\mu_y^*(\mathbf{x})$	1.000	0.997	0.96	0.98	1.96	−1.62	Unbiased
II	2-20-2	$\mathbf{x}$	$\sigma^{*2}(\mathbf{x})$	0.845	0.561	0.96	0.45	1.96	1.52	Unbiased
III	2-20-2	$\mathbf{x}$	$\mu_y^*(\mathbf{x})$	1.000	0.998	1.04	1.08	1.96	−0.61	Unbiased
III	2-20-2	$\mathbf{x}$	$\sigma^{*2}(\mathbf{x})$	0.900	0.856	0.96	0.75	1.96	6.20	Biased
Comparison with actual target (d)										
I	2-10-1	$\mathbf{x}$	$\mu_y^*(\mathbf{x})$	0.942	0.967	1.04	1.11	1.96	0.49	Unbiased
Comparison with actual target (d) and squared residuals										
II	2-20-2	$\mathbf{x}$	$\mu_y^*(\mathbf{x})$	0.939	0.965	1.04	1.00	1.96	−1.11	Unbiased
II	2-20-2	$\mathbf{x}$	$\sigma^{*2}(\mathbf{x})$	0.246	−0.743	1.04	1.29	1.96	1.16	Unbiased
III	2-20-2	$\mathbf{x}$	$\mu_y^*(\mathbf{x})$	0.942	0.967	1.04	1.11	1.96	−0.10	Unbiased
III	2-20-2	$\mathbf{x}$	$\sigma^{*2}(\mathbf{x})$	0.219	−0.627	1.04	1.51	1.96	4.39	Biased

The proposed network produces an unbiased estimate, $\mu_y^*(\mathbf{x})$, of both the underlying true regression function and the target d in all three training phases. The estimate $\sigma^{*2}(\mathbf{x})$ is also an unbiased estimate of both the true noise variance function and actual squared error for Phase II; the corresponding Phase III estimates are biased

As in Example #1 the correlation of the absolute errors with the estimated absolute errors and with the true noise standard deviation (Rows 5 and 6) improved slightly from Phase II to Phase III. The correlation results in Table 1.4 are of a similar order to those for Example #1 in Table 1.3. Rows 7 and 9 distribution results show the dispersion of the actually occurring absolute errors was greater than indicated by the estimated and true noise standard deviation results given in Rows 8 and 10. The decreased correlation of absolute values of residual errors with the known noise standard deviation (Row 6) suggested that Phase III training should be omitted in the more realistic multivariate setting for these data. This conclusion was supported by Table 1.5, where hypothesis tests for Example #2 showed that the Phase III estimate $\sigma^{*2}(\mathbf{x})$ was a biased estimate of both the true noise variance and the squared residuals.

The results of Tables 1.4 and 1.5 for Phase II training in Example #2 showed the proposed network produced an unbiased estimate $\sigma^{*2}(\mathbf{x})$ of the input dependent noise variance function $\sigma^2(t,m)$, which is a smooth function of time to maturity t and moneyness m. Moreover, $\sigma^{*2}(\mathbf{x})$ was also an unbiased estimate of the actually occurring residual errors for $\mu_y^*(\mathbf{x})$. These results suggested that given a set of unseen input variables for which there is no corresponding targets, the proposed network was capable of producing an unbiased estimate of a target $d(t,m)$, in this case synthetic option prices. An unbiased estimate of the underlying regression giving rise to the target data, and a corresponding (known) noise variance function, was also obtained. The unbiased estimate of both the mean of the target and the true noise variance function suggested that prediction intervals based upon the proposed network provided a good estimate of the 95% prediction intervals.

1.4.3.4 Example #3: Tests Using Actual Option Prices

Example #3 used the same data set as Example #2. However, actual observed market prices of options, corresponding to the input variables t, and m, were now used as targets in place of the synthetic prices created as targets for Example #2.

Table 1.6 reports the results for Example #3. In this case, the comparisons of the network estimates are with the actual target O and the actual squared residuals for the estimate $\mu_y^*(\mathbf{x})$ only. There is no known underlying regression and true noise variance function. The pattern for Example #2 was repeated. The Example #3 estimate for the mean of the target O was unbiased for all three training phases with high values for R^2 and Adj. R^2. The Phase II estimate $\sigma^{*2}(\mathbf{x})$ was an unbiased estimate of the actual squared residuals of $\mu_y^*(\mathbf{x})$, as indicated by the t-test results. However, the Phase III estimate was biased, like the corresponding Example #2 result. The R^2 and Adj. R^2 figures for the estimate $\sigma^{*2}(\mathbf{x})$ were better than the corresponding Example #2 results. These results suggested that the proposed network could produce unbiased estimates of both the mean and squared residuals of the target values, where those targets were actual observed option prices corresponding to unseen input variables. The biased estimate $\sigma^{*2}(\mathbf{x})$ obtained in Phase III is a further evidence that Phase III training is superfluous in the more

Table 1.6 Estimates by proposed network (Example #3)

	Layers	Input				F-stat		t-statistic		t-test
Phase	[Nodes]	$\mathbf{x} = [t, m]$	Outputs	R^2	Adj. R^2	F_{crit}(1 tail)	F_{calc}(0.05)	t_{crit}(2 tail)	t_{calc}(0.05)	[Bias]
Comparison with actual target (O)										
I	2-10-1	$\mathbf{x}$	$\mu_y^*(\mathbf{x})$	0.960	0.999	1.04	1.09	1.96	0.47	Unbiased
Comparison with actual target (O) and squared residuals										
II	2-20-2	$\mathbf{x}$	$\mu_y^*(\mathbf{x})$	0.932	0.972	1.04	1.13	1.96	−0.31	Unbiased
II	2-20-2	$\mathbf{x}$	$\sigma^{*2}(\mathbf{x})$	0.385	0.347	1.04	6.14	1.96	0.63	Unbiased
III	2-20-2	$\mathbf{x}$	$\mu_y^*(\mathbf{x})$	0.957	0.974	1.04	1.10	1.96	0.70	Unbiased
III	2-20-2	$\mathbf{x}$	$\sigma^{*2}(\mathbf{x})$	0.435	0.365	1.04	1.14	1.96	−6.59	Biased

The proposed network produces an unbiased estimate, $\mu_y^*(\mathbf{x})$, of the mean of the target O in all three training phases. The estimate $\sigma^{*2}(\mathbf{x})$ is also an unbiased estimate of the actual squared error for Phase II; the corresponding Phase III estimate is biased

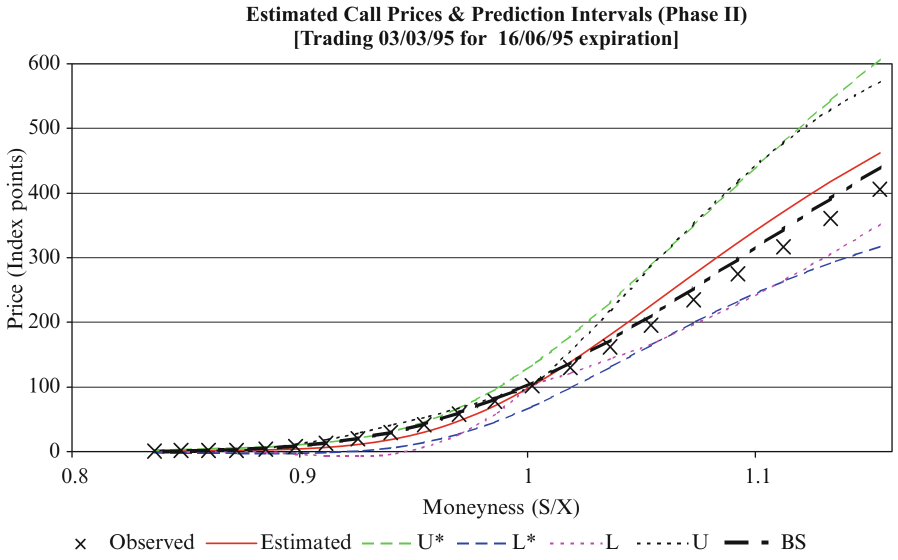


Fig. 1.2 Prediction intervals for Example #3. Prices of FT-SE100 call options trading on the 3rd March 1995 for June 1995 expiration. The *x*'s are observed option prices. Estimated is the network estimate of the prices. U^* and L^* are estimated upper and lower prediction intervals. U and L are corresponding intervals calculated using the actually occurring residual errors. BS is the Black–Scholes price prediction

realistic setting. Moreover, the estimate $\mu_y^*(\mathbf{x})$ is not improved in Phase III, as indicated by the poorer t-statistic. The unbiased Phase II estimates of the target mean, and the actually occurring squared residuals, suggested a good estimate of the 95% prediction intervals was given in this case also.

This is confirmed by inspection of Fig. 1.2 where the estimated prediction intervals (the blue dashed lines) correspond well with intervals calculated using the actually occurring residual errors (the green dashed lines). The option price series illustrated in Fig. 1.2 is new unseen data. The Example #3 model gives an unbiased price prediction for the series, with a t-statistic of -0.27 in an independent t-test assuming unequal variances. By comparison, the BS prediction of the option price is also unbiased for this series with a t-statistic of -0.20. In a paired t-test of the model predictions, the two models show no significant difference and the t-statistic is -1.12. In a paired t-test of the model residuals, a statistic of 1.12 is obtained. These results suggest that both the Example #3 model and the BS formula provide good models of the DGP for this sample. The relatively good result for the BS formula given the small sample of 21 observations is surprising, as the BS formula usually gives biased results for large samples. The Example #3 NN model though has only two inputs, moneyness and time to maturity, compared to the five inputs of the BS formula.

1.4.3.5 Example #4: Tests Using Actual Option Prices and Optimised Network

To facilitate equal comparison with the BS formula a further NN model, Example #4, was trained. Example #4 used the same training and test data used for Example #3. For Example #4, however, all five of the BS inputs were used. In addition, the network architecture was optimised, using sensitivity based pruning, to give four hidden layer nodes. Table 1.7 gives the results for Example #4.

In Table 1.7 the estimate $\mu_y^*(\mathbf{x})$ is again unbiased for all three phases. The fit to the observed option prices O is very good with Phases II and III R^2 and Adj. R^2 figures > 0.99, which is higher than the corresponding figures for Example #3. In addition, the F-test results for the estimate $\mu_y^*(\mathbf{x})$ indicate no significant difference in the variance compared with the target O. However, the Phase II estimate $\sigma^{*2}(\mathbf{x})$ is now biased, as well as the Phase III estimate. Inspection of the means of the actual squared residuals and their estimates given by $\sigma^{*2}(\mathbf{x})$ indicate an underestimate. This result suggests the proposed method underestimates the magnitude of squared residuals when the fit to the target O (or d) is very good. However, it is unlikely to be a problem in practice, as the underestimate only occurs when $\mu_y^*(\mathbf{x})$ is an unbiased estimate of the target O, and the fit is better than $R^2 > 0.99$, *and* calculated F-statistics are less than their critical values.[3]

Figure 1.3 shows the results of applying Example #4 to the same option price series illustrated in Fig. 1.2. In Fig. 1.3 the estimated prediction intervals (the blue dashed lines) correspond well with intervals calculated using the actually occurring residual errors (the green dashed lines), for the region of moneyness >1.05 where the fit of $\mu_y^*(\mathbf{x})$ to O is in general poorer than the region where moneyness is <1.05. However, the overall fit for Example #4 is much better than the fit for Example #3, which is itself unbiased. It can be seen that the BS price predictions are outside the prediction bands for the Example #4 predictions, for moneyness >1.0. Figures 1.2 and 1.3 graphically illustrate the performance of the proposed method for estimating prediction intervals and the utility of prediction intervals for assessing the differences in performance over the input space of option pricing models.

1.5 Conclusions

This chapter provides an exposition of methods for estimating confidence and prediction intervals on outputs, from computational intelligence tools used for data modelling. The underlying theory and the performance of a variety of

[3]The method is based on the use of a least squares cost function. However, maximum likelihood and least squares estimators are equivalent in terms of performance, and the tendency for maximum likelihood estimators to underestimate the variance is known, and has been remarked in [3] Chap. 6, Sect. 6.3.

Table 1.7 Estimates by proposed network (Example #4)

	Layers	Input				F-stat		t-statistic		t-test
Phase	[Nodes]	$\mathbf{x} = [S, X, t, r, iv]$	Outputs	R^2	Adj. R^2	F_{crit}(1 tail)	$F_{\text{calc}}(0.05)$	t_{crit}(2 tail)	$t_{\text{calc}}(0.05)$	[Bias]
Comparison with actual target (O)										
I	5-4-1	$\mathbf{x}$	$\mu_y^*(\mathbf{x})$	0.995	0.997	1.04	1.02	1.96	−0.61	Unbiased
Comparison with actual target (O) and squared residuals										
II	5-4-2	$\mathbf{x}$	$\mu_y^*(\mathbf{x})$	0.992	0.995	1.04	1.02	1.96	−0.44	Unbiased
II	5-4-2	$\mathbf{x}$	$\sigma^{*2}(\mathbf{x})$	0.294	0.071	1.04	258.32	1.96	4.81	Biased
III	5-4-2	$\mathbf{x}$	$\mu_y^*(\mathbf{x})$	0.990	0.994	1.04	1.04	1.96	−0.46	Unbiased
III	5-4-2	$\mathbf{x}$	$\sigma^{*2}(\mathbf{x})$	0.335	0.312	1.04	7.27	1.96	2.40	Biased

The proposed network produces an unbiased estimate, $\mu_y^*(\mathbf{x})$, of the mean of the target O in all three training phases. The estimate $\sigma^{*2}(\mathbf{x})$ is biased for both Phases II and III

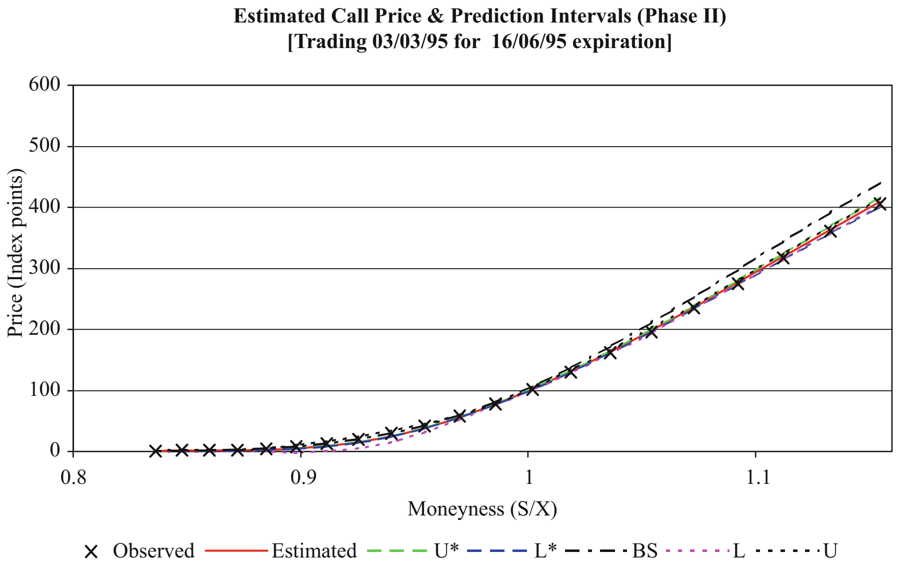


Fig. 1.3 Prediction intervals for Example #4. Prices of FT-SE100 call options trading on the 3rd March 1995 for June 1995 expiration. The *x*'s are observed option prices. Estimated is the network estimate of the prices. U^* and L^* are estimated upper and lower prediction intervals. U and L are corresponding intervals calculated using the actually occurring residual errors. BS is the Black–Scholes price prediction

different empirical applications of this theory are critically reviewed. Limitations of the existing approaches are outlined. A method for computing standard errors, confidence intervals, and prediction intervals which addresses these limitations is presented. This method is applicable to any sufficiently flexible computational intelligence technique used for non-linear regression. The method rests on the classical framework for least squares regression and maximum likelihood estimation. The implementation of this method was first described in [9], and it relies on a specific training algorithm. The training algorithm is easily adaptable to standard neural net software and a broad class of other computational intelligence techniques.

Test results for the method are presented here. It was applied successfully to a standard synthetic data set and gave statistically acceptable results. It performed comparably with the method of Nix and Weigend [19] in this test. A test using synthetic option prices was also performed and the true noise variance function successfully recovered. In a test with actual option prices, an unbiased estimate of the true squared errors for the fitted option prices was obtained. Further tests with actual option prices suggest the noise variance is underestimated when Adj. R^2 is greater than 0.99, but this is a feature of all methods based on least squares or maximum likelihood. The theory presented and the test results confirm that the method is suitable for use with the generality of financial market data.

References

1. E.O. Abensu, Genetic algorithms for development of new financial products. Braz. Rev. Finance **5**(1), 59–77 (2007)
2. C.M. Bishop, Mixture density networks. Technical report NCRG/4288, Neural Computing Research Group, Aston University (1994)
3. C.M. Bishop, *Neural Networks for Pattern Recognition* (Clarendon Press, Oxford, 1995)
4. C.M. Bishop, C.S. Qazaz, Bayesian inference of noise levels in regression. Technical report, Neural Computing Research Group, Aston University (1995)
5. C.M. Dahl, S. Hylleberg, Flexible regression models and relative forecast performance. Int. J. Forecast. **20**(2), 201–217 (2004)
6. B. Efron, R.J. Tibshirani, *An Introduction to the Bootstrap* (Chapman and Hall, New York, 1993)
7. F. Girosi, T. Poggio, Networks and the best approximation property. Biol. Cybern. **20**, 169–176 (1990)
8. J.M. Górriz, C.G. Puntonet, M. Salmern, J.J.G. de la Rosa, A new model for time-series forecasting using radial basis functions and exogenous data. Neural Comput. Appl. **13**(2), 101–111 (2004)
9. J.V. Healy, M. Dixon, B.J. Read, F.F. Cai, Confidence in data mining model predictions: a financial engineering application, in *Proceedings of the 29th Annual Conference IEEE Industrial Electronics Society*, vol. 2 (2003), pp. 1926–1931
10. T. Heskes, Practical confidence and prediction intervals, in *Advances in Neural Information Processing Systems*, vol. 9, ed. by M. Mozer, M. Jordan, T. Petsche (MIT, Cambridge, 1997), pp. 176–182
11. K. Hornik, Multilayer feedforward networks are universal approximators. Neural Network **2**(5), 359–366 (1989)
12. K. Hornik, M. Stinchcombe, H. White, Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. Neural Network **3**, 551–560 (1990)
13. P.J. Huber, The behavior of maximum likelihood estimation under nonstandard conditions, in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, ed. by L.M. LeCam, J. Neyman (University of California Press, CA, 1967), pp. 221–233
14. J. Hutchinson, A.W. Lo, T. Poggio, A non-parametric approach to pricing and hedging derivative securities via learning networks. J. Finance **49**(3), 851–889 (1994)
15. J.T.G. Hwang, A.A. Ding, Prediction intervals for artificial neural networks. J. Am. Stat. Assoc. **92**(438), 748–757 (1997)
16. A. LeBaron, A. Weigend, Evaluating neural network predictors by bootstrapping, in *Proceedings of the International Conference on Neural Information Processing (ICONIP'94)*, Seoul, Korea, 1994, pp. 1207–1212
17. D.J.C. MacKay, Bayesian Methods for Adaptive Models. PhD thesis, California Institute of Technology (1991)
18. M. Maruyama, F. Girosi, T. Poggio, A connection between GRBF and MLP. Artificial Intelligence Memo 1291, Massachusetts Institute of Technology (1991)
19. D.A. Nix, A.S. Weigend, Learning local error bars for non-linear regression, in *Proceedings of NIPS 7*, 1995, pp. 489–496
20. C. Satchwell, Neural networks for stochastic problems: More than one outcome for the input space, in *NCAF Conference*, Aston University, 1994
21. G.A.F. Seber, C.J. Wild, *Nonlinear Regression* (Wiley, New York, 1989)
22. A. Skabar, Direction-of-change financial time series forecasting using neural networks: a Bayesian approach, in *Advances in Electrical Engineering and Computational Science*, ed. by S-I. Ao, L. Gelman. Lecture Notes in Electrical Engineering, vol. 39 (Springer, Berlin, 2009), pp. 515–524
23. R.L. Thomas, *Modern Econometrics* (Addison Wesley, MA, 1997)

24. R. Tibshirani, A comparison of some error estimates for neural network models. Neural Comput. **8**, 152–163 (1996)
25. L.H. Ungar, R.D. De Veaux, E. Rosengarten, *Estimating Prediction Intervals for Artificial Neural Networks* (University of Pennsylvania, Philadelphia, 1995)
26. C.J.F. Wu, Jacknife, bootstrap and other resampling methods in regression analysis. Ann. Stat. **14**, 1261–1295 (1986)
27. M. Zhu, L. Wang, Intelligent trading using support vector regression and multilayer perceptrons optimized with genetic algorithms, in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2010, pp. 1–5

Chapter 2
Can Artificial Traders Learn and Err Like Human Traders? A New Direction for Computational Intelligence in Behavioral Finance

Shu-Heng Chen, Kuo-Chuan Shih, and Chung-Ching Tai

Abstract The microstructure of markets involves not only human traders' learning and erring processes but also their heterogeneity. Much of this part has not been taken into account in the agent-based artificial markets, despite the fact that various computational intelligence tools have been applied to artificial-agent modeling. One possible reason for this little progress is due to the lack of good-quality data by which the learning and erring patterns of human traders can be easily archived and analyzed. In this chapter, we take a pioneering step in this direction by, first, conducting double auction market experiments and obtaining a dataset involving about 165 human traders. The controlled laboratory setting then enables us to anchor the observing trading behavior of human traders to a benchmark (a global optimum) and to develop a learning index by which the learning and erring patterns can be better studied, in particular, in light of traders' personal attributes, such as their cognitive capacity and personality. The behavior of artificial traders driven by genetic programming (GP) is also studied in parallel to human traders; however, how to represent the observed heterogeneity using GP remains a challenging issue.

S.-H. Chen (✉) • K.-C. Shih
AIECON Research Center, Department of Economics, National Chengchi University, Taiwan
e-mail: chen.shuheng@gmail.com; melvinshih@gmail.com

C.-C. Tai
Department of Economics, Tunghai University, Taiwan
e-mail: chungching.tai@gmail.com

2.1 Introduction and Motivation

2.1.1 Learning About Human Traders' Learning

When human subjects are placed in the market for trading competition [50, 51, 55], we have to admit that, up to the present, we do not have a good theory or even well-archived empirical evidence which can help answer the very general question regarding *who learns what and when*? The traditional approach to handling this issue is very much in the line initiated by Arthur [4], which is to compare the patterns observed from human traders with those observed from artificial (machine-learning) agents and, based on the similarity of the pattern, to decide whether human-subject learning has been well captured by the proposed computational intelligence models, such as genetic algorithms [2,3], reinforcement learning [20], and so on and so forth.

One, of course, can gain some insights from this *mirroring approach* [16] that is conditional on a carefully chosen similarity metric. However, saying that human traders behave like the artificial agents, driven by evolutionary computation or reinforcement learning, seems at best to be only a *first-order approximation* of many complex or complicated details that human traders may face in the real markets, but are nonetheless difficult model at this stage. Humans are emotional beings and have different personal traits, which can easily result in great deviations from the behavioral dynamics as predicted by computational intelligence, be they genetic algorithms or reinforcement learning.

We try to use Fig. 2.1 to elaborate on this point. As typically assumed in most textbooks on financial mathematics, a global optimum exists for a well-defined trading problem. For example, it can be an optimum trading strategy which advises traders with respect to both the market timing and pricing (bids or asks) decisions. Given the existence of the global optimum, presumably one can then define and measure errors that a trader made based on the observed deviations. Both artificial traders and human traders can make mistakes and mistakes may have their *patterns*.[1] These patterns can be further analyzed to understand the underlying mechanisms which cause these patterns. In addition, in response to the errors, both artificial traders and human traders are supposed to learn, and their learning may also have patterns.[2] The question is then whether one can use computational intelligence to construct artificial agents in a way that both patterns of errors and learning observed from human subjects can be well understood.

What will be claimed in this chapter is that studies devoted to these issues are still in their infancy stage. While studies devoted to the financial application of

[1]Various patterns of mistakes, also known as *behavioral biases*, have been long studied by psychologists and social scientists. See [5], Part II, for a review of various biases. Also see [34].

[2]Learning does not necessarily mean correction in a right direction; the well-known over-reaction or over-adjustment are typical examples of this pattern of learning [48]. Furthermore, learning may take a while to see its effect; this is known as slower learning or the inertial effect [12].

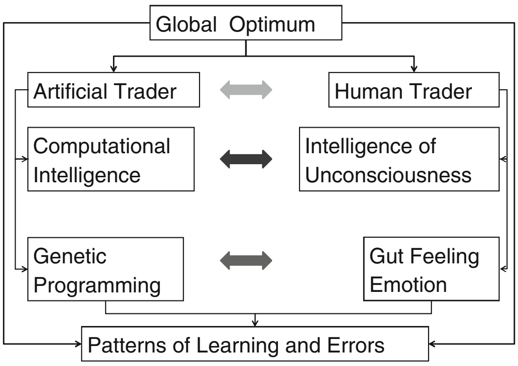

Global Optimum
Artificial Trader
Human Trader
Computational Intelligence
Intelligence of Unconsciousness
Genetic Programming
Gut Feeling Emotion
Patterns of Learning and Errors

Fig. 2.1 Artificial traders and human traders

computational intelligence are piling-up research, most of them only have artificial traders or programmed traders as their main concern (the left part of Fig. 2.1). Few ever go further to see the possible connections to real human traders. Although the term "heuristics" is a psychology-oriented term and has lately also been widely used in computational intelligence,[3] the heuristics developed in the latter tend to be very much disentangled from the former.[4] It seems to us that the former belongs to a separate literature (the right part of Fig. 2.1) known as behavioral finance, psychological finance or, recently, neurofinance [5].

The tools and the languages used by intelligent finance and psychological finance are very different. For the former, the decision is made based on intensive search and data mining, such as genetic programming, whereas for the latter the decision is often made by very limited search in a very spontaneous and reflexive manner, such as gut feelings [32]. Needless to say, to build human-like artificial traders, it would be necessary to narrow the gap.[5] Hence, the first step is to have a thorough understanding of what kinds of patterns, both in learning and error-making, are neglected by the conventional financial applications of computational intelligence [18].

2.1.2 *Research Framework*

In this study, we bring a new direction by taking a first step in narrowing the gap. Instead of fitting a specific CI model to the observations of human traders, we propose a sensible measure, called the *learning index*, which takes into account some important details and hence sheds light on the exact cognitive orientation of human traders in the bazaar.

[3]See [33] for a simple historical review of the use of this term.

[4]For example, the recently published handbook on metaheuristics [31] has no single mention of psychology.

[5]Probably partially because of this gap, artificial traders cannot replace human traders [15].

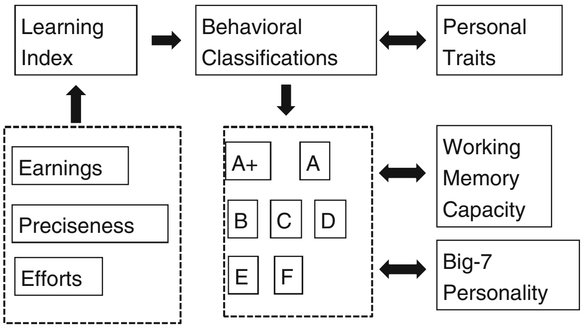


Fig. 2.2 Research framework

Our proposed learning index is based on three major elements related to the behavior of human traders. These three elements are *earning capacity*, *trading preciseness*, and *trading efforts*, as shown in the left block of Fig. 2.2. Each of the three will be motivated and detailed later; they together show the distinguishing feature of this learning index: not only can each element tell us whether the human agents have learned, but more importantly can inform us of the *quality* of their learning, which includes the degree, speed, and stability (fragility) of their learning. Hence, it gives us not just a one-shot end-result but more on the *process*, and, as we mention above, it is the process that matters in the applications of computational intelligence to modeling the human-trader behavior.

We then use the learning index to classify the performance of human subjects into distinctive groups, which basically range from inferior learning to superior learning. Hence, at the low end, such as Class "F" (middle block, Fig. 2.2), we have human subjects who have learned little or not at all, whereas, at the very top end such as "A+" or "A" (middle block, Fig. 2.2), we have subjects who have learned by heart. In the middle, we have subjects whose learning is not complete and their confidence about what they have learned has not been established either. For them, while the sky is not entirely clouded, shadows appear here and there.[6]

We then can proceed further to understand the causes of the observed heterogeneity among different subjects. The observed heterogeneities of human agents have not been represented by the standard applications of computational intelligence, and hence the causes of the observed heterogeneity have generally been neglected in the literature on artificial agents [18]. The study has not been picked up until very recently [19]. While human subjects can be heterogeneous in many dimensions, this chapter is limited to only two important ones: *cognitive capacity* and *personality*. These two dimensions are included because the literature indicates that they, by and large, can have an impact upon the decision-making quality [14, 46]. Hence, as a first step, we would like to examine their contribution to account for the observed heterogeneity in learning.

[6]Recently, there have been a number of studies focusing on the neurocognitive study of decision making under *uncertainty* or *ambiguity*, which may well serve as a neural foundation for the observed behavioral phenomena here [36, 54].

The proposed research framework with the three major components is summarized in Fig. 2.2. To illustrate the implementation of this framework, below we shall provide a concrete example based on the *double auction markets*. However, before we proceed further, let us wrap up this section by pointing out that each component of the proposed framework is flexible enough to make it adaptable to different applications. The essence is to meaningfully understand the learning and erring processes of human traders in a controlled (experimental) environment and hence to bridge the gap in learning and erring behavior between artificial traders and human traders, if the latter, to a quite large extent, cannot be replaced by the former [15].

The rest of this chapter is organized as follows. Section 2.2 describes the trading environment, a double auction market, based on which the laboratory experiments were designed. The global optimal trading strategy in this trading environment can be derived as a solution from a combinatorial optimization problem (integer programming). The solution can be read as an application of the economic theory of optimal procrastination. With this global optimum, Sect. 2.3 proposes the learning index which can help us observe the learning and erring patterns of both artificial traders and human traders. It can further help cluster different behavioral patterns, upon which the optimum-discovery capability of human traders can be observed. Section 2.4 applies the established learning index to sets of 165 and 168 human traders, respectively, and then associates the observed heterogeneities among these traders with their personal attributes, including cognitive capacity and personality. Section 2.5 presents the concluding remarks.

2.2 Trading Environment: The Double Auction Markets

In this study, both artificial traders and human traders are placed in a typical double auction market experiment [53]. In a double auction market, both buyers and sellers can submit bids and asks. This contrasts with only buyers shouting bids (as in an *English Auction*) or only sellers shouting asks (as in a *Dutch Auction*). There are several variations of DA markets. One example is the *clearinghouse* DA of the Santa Fe Token Exchange (SFTE) [50] on which this work is based.

On the SFTE platform, time is *discretized* into alternating *bid/ask* (BA) and *buy/sell* (BS) steps. Initially, the DA market opens with a BA step in which all traders are allowed to simultaneously post bids and asks for one token only. After the clearinghouse informs the traders of each others' bids and asks, the holders of the *highest bid* and *lowest ask* are matched and enter a BS step. During the BS step, the two matched traders carry out the transaction using the *mid-point* between the *highest bid* and the *lowest ask* as the transaction price. Once the transaction is cleared, the market enters a BA stage for the next auction round. The DA market operations are a series of alternating BA and BS steps.

The specific market architecture employed in this study has four buyers and four sellers. They are numbered from Buyer 1 to Buyer 4 and Seller 1 to Seller 4,

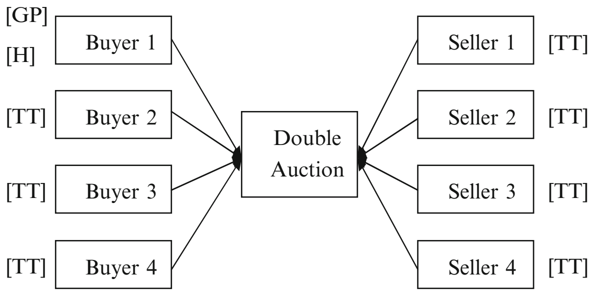


Fig. 2.3 Composition of market participants

Table 2.1 The token value table

	Token 1	Token 2	Token 3	Token 4
Buyer 1	10,518	10,073	6,984	6,593
Buyer 2	10,519	10,072	6,981	6,593
Buyer 3	10,516	10,071	6,985	6,589
Buyer 4	10,521	10,071	6,987	6,590
Seller 1	622	1,013	4,102	4,547
Seller 2	622	1,010	4,101	4,548
Seller 3	618	1,014	4,100	4,545
Seller 4	619	1,016	4,100	4,550

accordingly, as shown in Fig. 2.3. The commodity traded in this market is called the *token*. Buyers value these tokens and their *maximum willingness to pay* (the *reservation price* of buyers) for each token is specified in the *token-value table*. The willingness to pay is nonincreasing with the number of tokens already owned. For example, in Table 2.1, for Buyer 1, the maximum willingness to pay for the first token is 10,518, followed by 10,073 for the second, 6,984 for the third, and 6,593 for the fourth. On the other hand, sellers would like to provide these tokens and the *minimum acceptable price* (the *reservation price* of the seller) for each token is also specified in the token-value table. As the opposite of the maximum willingness to pay, the minimum acceptable price is nondecreasing with the number of tokens already sold. Let us use Seller 1 in Table 2.1 as an example. The minimum acceptable price starts with 622 for the first token, then 1,013 for the second, 4,102 for the third, and 4,547 for the fourth.

This structure of the token-value table is generated in light of the familiar behavior of marginal utility and marginal cost and hence it fits well with the law of demand and supply. If we pool all of the maximum willingness to pay and the minimum acceptable price together, and arrange them in descending order and ascending order separately, we can then draw a downward-sloping demand schedule and upward-sloping supply schedule, as shown in Fig. 2.4.

The artificial traders and human traders, placed in this market environment, will play the role of Buyer 1 (Fig. 2.3). The trading behavior of the artificial traders and the human traders is the focus of this study. The artificial traders will be programmed

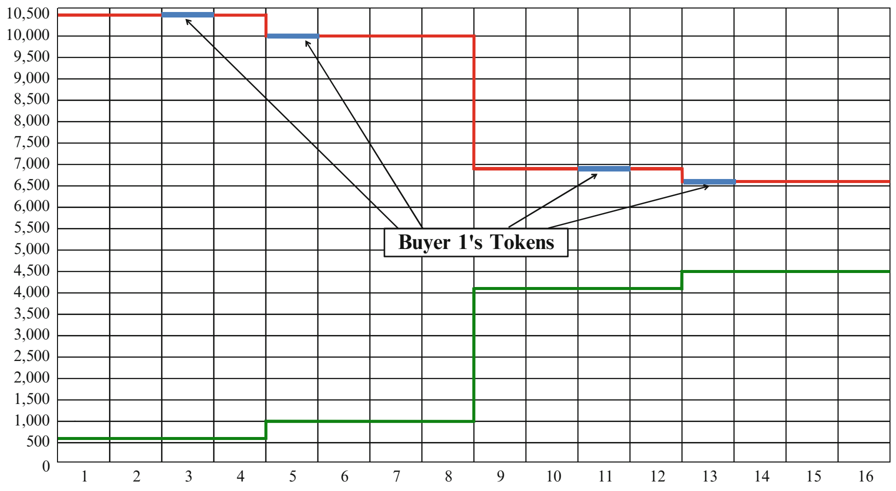


Fig. 2.4 The demand–supply schedule: the demand and supply schedule given above is drawn by arranging a list of all reservation prices in descending order for buyers' reservation prices and in ascending order for sellers' reservation prices

by *genetic programming*.[7] We also assume that all other market participants, i.e., the opponents of the artificial or human traders, are *truth tellers* (Fig. 2.3). Being a truth teller, the trader simply bids or asks at his current reservation price (along the demand and supply curves).

2.2.1 Benchmark: Market Timing and Bids

The institutional assumption of a discrete-time double auction in the form of SFTE coupled with the behavioral assumption of truth telling enables us to represent the given environment as a solvable *combinatorial optimization problem* (see the Appendix) [56]. Solving this optimization problem will give Buyer 1 the best *market timing* and the *most favorable bids*, which together leads to the highest trading profits for Buyer 1. The solvability of this problem allows us to keep a *benchmark* upon which our further analysis of learning and erring is based (Figs. 2.1 and 2.2).

For Buyer 1, the unique optimal trading strategy for the constrained combinatorial optimization is derived and presented in the left panel of Table 2.2. The panel has three columns, and the leftmost one is simply the number of trading steps, and, by following SFTE, there are a total of 25 trading steps for each market experiment.

[7]Chen [17] argues that genetic programming equips economists with a tool to model the chance-discovering agent, which is an essential element of modern economic theory.

Table 2.2 Trading schedule by optimization (*left panel*) and by the GP trader (*right panel*)

Optimization			GP simulation		
Step	Bidding time and bid	Match or not	Step	Bidding time and bid	Match or not
1	−1		1	−1	
2	−1		2	618	
3	−1		3	619	
4	−1		4	622	
5	−1		5	622	
6	−1		6	1,010	
7	6,988	Yes	7	1,013	
8	6,988	Yes	8	1,014	
9	−1		9	1,016	
10	−1		10	4,100	
11	−1		11	4,100	
12	−1		12	4,101	
13	−1		13	4,102	
14	−1		14	4,545	Yes
15	4,548	Yes	15	4,545	
16	4,550	Yes	16	4,547	Yes
17	−1		17	4,547	
18	−1		18	4,548	Yes
19	−1		19	4,548	
20	−1		20	4,550	Yes
21	−1		21	−1	
22	−1		22	−1	
23	−1		23	−1	
24	−1		24	−1	
25	−1		25	−1	

All trades have to be finished within these 25 steps. Normally, this is more than what an optimal trading strategy needs. The question is when to get into the market and how much to bid (ask), and the answers are shown in the 2nd column. Hence we can see that in this specific market, the optimal time to enter the market is at steps 7, 8, 15, and 16 with bids of 6,988, 6,988, 4,548, and 4,550, respectively. In this way, Buyer 1 can earn a maximum profit of 17,067. The symbol "−1" also showing in many rows of the table is not a value to bid, but a sign indicating "Pass," i.e., not entering the market. The third column then shows whether a deal is made given the offers (bids) and the asks from the sellers. Being an optimal trading strategy, this means that all bids are successfully matched, as the sign "Yes" indicates.

The intuition behind this optimum solution is the economic theory of *optimal procrastination*, which basically means that the trader attempts to delay his participation in the market transaction so as to avoid early competition and become a *monopsonist* in the later stage. Once getting there, he will then fully exercise the monopsony power by bidding with *third-degree price discrimination*. However, procrastination may also cause the agent to miss some good offers; therefore, there is an opportunity cost for procrastination and the agent will try to optimize the procrastination time by balancing his monopsony profits against these costs. Procrastinating in *two* stages gives the balance. As shown in Fig. 2.4, there is a sharp fall in the market demand curve accompanied by the sharp rise in the supply curve, which suggests dividing the sequence of trading actions into two, one before the change and one after the change.

2.2.2 Deviations: A Case of the GP Trader

The benchmark is an optimal strategy. In that sense, it becomes a rest state; additional efforts for searching or learning are not necessary. Therefore, as a benchmark, it helps us not only evaluate the earning performance of GP, but also enables us to see how much energy is being devoted to searching and learning. The second kind of deviation, deviation from *effort minimization*, can be equally, if not more, important than the first kind of deviation, deviation from *profit maximization*, even though in the machine learning literature we often only consider the first kind rather than the second.

Let us illustrate the second kind of deviation using one example from GP, as shown in Table 2.2.[8] This specific trading strategy found by GP generates the following trading behavior. We notice that, compared to the benchmark, GP traded at a different time schedule and made deals in periods 14, 16, 18 and 20 with lower bids (from 4,445 to 4,550). Not surprisingly, in this way, it also ended up with a lower profit of 15,978, rather than the maximum one of 17,067.

In light of the benchmark, we can see that the GP trader also learned to delay trading (as clearly shown in Table 2.2), but it did not procrastinate in the optimal way, i.e., in two stages; instead, it did so in one stage only. In addition to that, the GP trader had a total of 19 visits to the market, which is four times higher than the minimum effort required by the benchmark, i.e., four visits only. Hence, in sum, the GP traders deviate from the benchmark in *earnings*, *trading schedule* (market timing and bids), and *trading frequency*. The interpretation of these deviations for artificial traders, like GP, can be very different from that for human traders. All kinds of feelings related to uncertainty, such as gut feelings, fast and frugal heuristics, greed, fear, regret aversion, risk aversion, fatigue, and overconfidence, good or bad, may cause human traders to deviate from the benchmark in all three above-mentioned dimensions, and probably, in very different manner, too [52].

Alternatively put, the learning and erring behavior of human traders should be studied in the context of both cognitive psychology and personality psychology [30, 43–45]. Unless these psychological attributes have been incorporated into the design of artificial agents, one could hardly expect the same deviation patterns

[8]The details of the GP run in this chapter can be found in [22].

between humans and machines.[9] Therefore, in the following, we will propose a measure of learning or a learning index which not only allows us to examine the end-results (earnings), but also enables us to trace some *psychological details* of the learning process of human subjects.

2.3 Leaning Index

2.3.1 The Three Criteria

What is proposed in this section is a learning index (LI) built upon the three above-mentioned possible deviations, namely, earnings, the trading schedule (market timing and bids), and trading frequency. The basic idea is to assign credits to the three above-mentioned criteria in such a way that in the end very different behavioral patterns can be easily distinguished. This is summarized in Table 2.3.

Here, very much in the spirit of the widely used *balanced scorecard* [40], we assign credits for each strategy or action that either fulfills or partially fulfills the target (given by the benchmark). Hence, X_1 points are given to the action if it leads to the maximum profit, and X_2 if it fails to do so. Obviously, $X_1 > X_2$. As we have learned from Table 2.2, if the action fails to fulfill the target return, it must then fail to follow the target trading schedule in terms of either market timing or bidding. Hence, a partial credit of Y points will be given for each single successful match. From Table 2.2, we know that these are a total of four units to trade; therefore, the trader will be assigned $4 \times Y$ points if the entire trading schedule is matched. Otherwise, it could be $3 \times Y, 2 \times Y, \ldots,$ all the way down to zero if there is no single match.

It is possible that the trader can still gain the maximum profits while not following the target trading schedule, because not all trades will be successful or effective, as we have seen from the GP trader in Table 2.2. Therefore, in considering that each offer, regardless of being successful or not, involves a cost, broadly known as the

Table 2.3 Credit assignment rule

	Criterion	Credits assigned	Range
1	Target earning	X_1 points if achieved X_2 points if not achieved X_3 points if surpassed	X_2, X_1, X_3
2	Trading schedule	Y for every single match	$0, Y, 2Y, 3Y, 4Y$
3	Trading frequency	Z for each entering to the market	$21Z, 20Z, \ldots, 0$

[9]The fundamental pursuit here is: when a mistake is made, what are the differences between that made by an artificial agent and that made by human agents?

transaction cost,[10] we “credit” each additional unnecessary entrance to the market with Z points and $Z < 0$. Since a single market experiment lasts for 25 steps and 4 steps are necessary for finishing all possible trades, the trader can be “credited” with $21 \times Z$ as a maximum for his transaction cost.

2.3.1.1 Greed and Gambling

This gives the basic structure of the learning index. One thing which enables us to make a further distinction is the case where the trader may earn a profit which is higher than the benchmark. This subtle situation can occur when there is a piece of luck on which the benchmark does not rely on. This kind of luck occurs when a deal can be made with two or more identical offers, say, two identical bids, and then a lottery will be applied to decide who has the right to buy.

For example, according to the benchmark, the second token shall be bid in step 8 at a price of 6,988 (Table 2.2), which is one dollar higher than what Buyer 4 will bid at that moment (Table 2.1). Hence, if instead of 6,988 Buyer 1 bids more *greedily* also with 6,987, he may still get the deal with a 50% chance. If he has that luck, he may even earn an additional profit of 0.5, up to a total of 17,067.5. Nevertheless, if it is Buyer 3 who has the luck and not him, then he will lose the good price, 1,016, offered by Seller 4, and the next available quote for him will be a much higher 4,100, offered by Seller 3 or 4 (Fig. 2.4 and Table 2.1), which can cause him a dramatic drop in profits, from the target 17,067 to 16,622. Since our benchmark will not take this risky action, it may, therefore, lose to a trader who would like to bet on this luck.

This delicate design enables us to observe human traders’ exploration of profit opportunities and their reevaluation of these opportunities after being aware of the underlying risk. We call this process “route to a gambler,” and we wish to examine how traders’ personalities may have an effect on the choice of this route and how this route has affected traders’ total performance. To achieve this goal, a credit of X_3 will be assigned to the *gambler* ($X_3 > X_2$) if he did conclude a successful deal (Table 2.3). In this way, we can easily identify the occurrence of the gambler’s route during the trader’s learning process.

2.3.2 Illustrations

The proposed learning index is illustrated with three cases, one GP trader and two human traders. Before we do so, we need to set the values of the credit parameters appearing in Table 2.2. There is no unique way to set these values. Many possible sets of values should work fine as long as they help us easily separate agents with

[10]It does not have to be narrowly limited to the pecuniary costs associated with trading, such as broker fees or the Tobin tax. It can cost personal health as well [6].

Table 2.4 Parameter setting for the credit assignment

Parameter	Value
X_1	1,000
X_2	0
X_3	2,000
Y	100
Z	-1

very different leaning and erring patterns. In a sense, these sets of values serve the role of separating the hyperplane, such as the support vector machine. With this understanding, we, therefore, arbitrarily choose one set of values, as given in Table 2.4, and will fix this setting throughout the rest of the chapter.

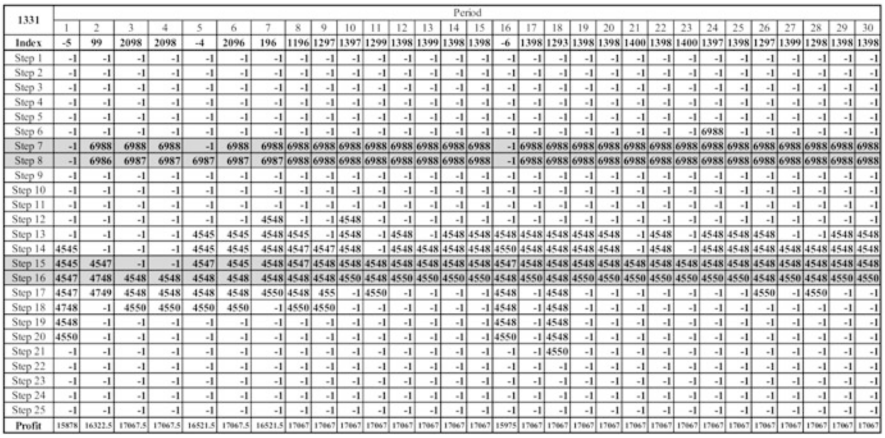
2.3.2.1 Artificial Trader

We begin with a very simple demonstration and apply the learning index to the GP trader introduced in Table 2.2. First, this GP trader did not earn the target profit; hence, his earning performance is credit zero ($X_2 = 0$). Second, he failed to follow the target trading schedule from the first token to the last token; hence, the credit assigned to his trading schedule is also zero ($0 \times Y = 0$). Finally, as to the transaction frequency, he also failed to use the necessary trading times: he used 15 more times. His credits earned in this part become -15 ($15 \times Z = -15$). As a total, the learning index of this GP trader is, therefore, -15 ($= X_2 + (0 \times Y) + (15 \times Z)$).

2.3.2.2 Human Traders

Human-subject experiments in double auction markets were conducted at the Experimental Economics Laboratory (EEL), National Chengchi University, from May to July 2010. Each student played the role of Buyer 1 as shown in Fig. 2.4. The DA experiments were repeated 30 times for each subject, and as a whole could be finished in 1 h. In addition to the DA experiment, they were also paid to run two additional psychological tests, namely, the *working memory test* (Sect. 2.4.2.1) and *personality test*. The experimental results, including their DA market performance as well as psychological tests, are all well archived in the Experimental Subject Database (ESD). From this database, we successfully retrieved a set of 165 subjects with the working memory test and 168 subjects with the personality test.[11] In the following, our illustration will be based on two representative subjects, namely, Subjects 1331 and 1129.

[11]In fact, there are a total of 185 subjects attending the double auction experiments, but for some of them the data are incomplete. Hence, for the WMC test the valid sample has 165 subjects, and for the personality test the valid sample has 168 subjects. There are 151 subjects appearing in both samples.



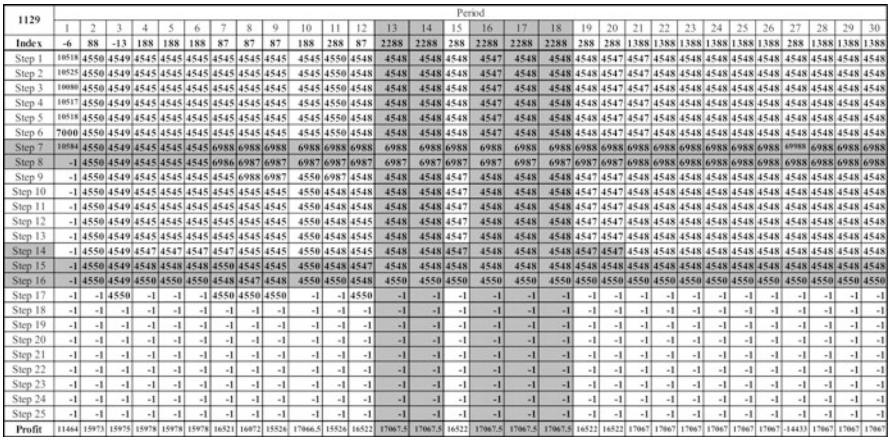
1331	Period																													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Index	-5	99	2098	2098	-4	2096	196	1196	1297	1397	1299	1398	1399	1398	1398	-6	1398	1293	1398	1398	1400	1398	1400	1397	1398	1297	1399	1298	1398	1398
Step 1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 3	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 5	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 6	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	6988	-1	-1	-1	-1	-1	-1
Step 7	-1	6988	6988	6988	-1	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	-1	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988
Step 8	-1	6986	6987	6987	6987	6987	6987	6988	6988	6988	6988	6988	6988	6988	6988	-1	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988
Step 9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 10	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 11	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 12	-1	-1	-1	-1	-1	-1	4548	-1	-1	4548	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 13	-1	-1	-1	-1	4545	4545	4548	4545	-1	4548	-1	4548	-1	4548	4548	4548	4548	4548	4548	4548	-1	4548	-1	4548	4548	4548	-1	-1	4548	4548
Step 14	4545	-1	-1	-1	4545	4545	4548	4547	4547	4548	-1	4548	4548	4548	4548	4550	4548	4548	4548	4548	-1	4548	-1	4548	4548	4548	4548	4548	4548	4548
Step 15	4545	4547	-1	-1	4547	4545	4548	4547	4548	4548	4548	4548	4548	4548	4548	4547	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548
Step 16	4547	4748	4548	4548	4548	4548	4548	4548	4548	4550	4548	4550	4550	4550	4550	4548	4550	4548	4550	4550	4550	4550	4550	4550	4550	4548	4550	4548	4550	4550
Step 17	4547	4749	4548	4548	4548	4548	4550	4548	455	-1	4550	-1	-1	-1	-1	4548	-1	4548	-1	-1	-1	-1	-1	-1	-1	4550	-1	4550	-1	-1
Step 18	4748	-1	4550	4550	4550	4550	-1	4550	4550	-1	-1	-1	-1	-1	-1	4548	-1	4548	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 19	4548	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	4548	-1	4548	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 20	4550	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	4550	-1	4548	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 21	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	4550	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 22	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 23	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 24	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 25	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Profit	15878	16322.5	17067.5	17067.5	16521.5	17067.5	16521.5	17067	17067	17067	17067	17067	17067	17067	17067	15975	17067	17067	17067	17067	17067	17067	17067	17067	17067	17067	17067	17067	17067	17067

Fig. 2.5 The learning and erring process of Subject 1331

Figure 2.5 gives the trading process of Subject 1331. As said, the DA experiments were repeated 30 times (periods). For each period, bids and asks were made and matched 25 times (steps). Hence, basically, what is demonstrated in Fig. 2.5 is a 25-by-30 table. Each column vector then indicates how the subject bid and made deals during the respective period. At the very top of the table, the "Index" row (the 2nd row) gives the sum of the credits assigned for each of the three criteria.

For example, this number in the 31st column (the last trading period) is 1,398, which can be broken down into 1,000 (X_1), 400 ($4 \times Y$), and -2 ($2 \times Z$). The reasons for these assigned credits are clear. The subject did earn the target profit (1,000 credits); in addition, the trading schedules of all four tokens (bidding time and bids) were exactly the same as the benchmark strategy (400 credits). Nevertheless, he also made two unnecessary early bids for the last two tokens; by the third criterion, he lost two points (-2). Therefore, his learning index (LI) over the three criteria is 1,398 points. This performance, compared to his initial value, -5 in period 1 and 98 in period 2, shows a significant improvement.

We, however, would like to draw readers' attention to two stylized patterns of human learning and erring. First, while many subjects are able to show significant improvement made over time, *their learning curve is not monotonically increasing and may fluctuate significantly*, which leads to our next point. The fluctuating pattern of their performance can be attributed to either *accidents* or *a lack of confidence* in what they learned. In the case of Subject 1331, we can see the relevance of these two possibilities. The sudden drop from a peak of "1,398" in period 15 to "-6" in period 16 could be hypothesized as an accident due to an absent mind (forgot to bid in Steps 7 and 8). In addition, in periods 21 and 23, the subject already had a full score; however, he was constantly trying something else nearby and that cost him some additional points. This indicates that some degree of uncertainty or confusion



1129	Period																													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Index	-6	88	-13	188	188	188	87	87	87	188	288	87	2288	2288	288	2288	2288	2288	288	288	1388	1388	1388	1388	1388	1388	288	1388	1388	1388
Step 1	10518	4550	4549	4545	4545	4545	4545	4545	4545	4545	4550	4548	4548	4548	4548	4547	4548	4548	4548	4547	4547	4548	4548	4548	4548	4548	4548	4548	4548	4548
Step 2	10525	4550	4549	4545	4545	4545	4545	4545	4545	4545	4550	4548	4548	4548	4548	4547	4548	4548	4548	4547	4547	4548	4548	4548	4548	4548	4548	4548	4548	4548
Step 3	10080	4550	4549	4545	4545	4545	4545	4545	4545	4545	4550	4548	4548	4548	4548	4547	4548	4548	4548	4547	4547	4548	4548	4548	4548	4548	4548	4548	4548	4548
Step 4	10517	4550	4549	4545	4545	4545	4545	4545	4545	4545	4550	4548	4548	4548	4548	4547	4548	4548	4548	4547	4547	4548	4548	4548	4548	4548	4548	4548	4548	4548
Step 5	10518	4550	4549	4545	4545	4545	4545	4545	4545	4545	4550	4548	4548	4548	4548	4547	4548	4548	4548	4547	4547	4548	4548	4548	4548	4548	4548	4548	4548	4548
Step 6	7000	4550	4549	4545	4545	4545	4545	4545	4545	4545	4550	4548	4548	4548	4548	4547	4548	4548	4548	4547	4547	4548	4548	4548	4548	4548	4548	4548	4548	4548
Step 7	10584	4550	4549	4545	4545	4545	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988	69988	6988	6988	6988
Step 8	-1	4550	4549	4545	4545	4545	6986	6987	6987	6987	6987	6987	6987	6987	6987	6987	6987	6987	6987	6987	6988	6988	6988	6988	6988	6988	6988	6988	6988	6988
Step 9	-1	4550	4549	4545	4545	4545	4545	6988	6987	4550	6987	4548	4548	4548	4547	4548	4548	4548	4547	4547	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548
Step 10	-1	4550	4549	4545	4545	4545	4545	4545	4545	4550	4548	4548	4548	4548	4547	4548	4548	4548	4547	4547	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548
Step 11	-1	4550	4549	4545	4545	4545	4545	4545	4545	4550	4548	4548	4548	4548	4547	4548	4548	4548	4547	4547	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548
Step 12	-1	4550	4549	4545	4545	4545	4545	4545	4545	4550	4548	4545	4548	4548	4547	4548	4548	4548	4547	4547	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548
Step 13	-1	4550	4549	4545	4545	4545	4545	4545	4545	4550	4548	4545	4548	4548	4547	4548	4548	4548	4547	4547	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548
Step 14	-1	4550	4549	4547	4547	4547	4547	4545	4545	4550	4548	4545	4548	4548	4547	4548	4548	4548	4547	4547	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548
Step 15	-1	4550	4549	4548	4548	4548	4550	4545	4545	4550	4548	4547	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548	4548
Step 16	-1	4550	4549	4550	4550	4550	4548	4547	4548	4550	4550	4548	4550	4550	4550	4550	4550	4550	4550	4550	4550	4550	4550	4550	4550	4550	4550	4550	4550	4550
Step 17	-1	-1	4550	-1	-1	-1	4550	4550	4550	-1	-1	4550	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 18	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 19	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 20	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 21	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 22	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 23	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 24	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Step 25	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Profit	11464	15973	15975	15978	15978	15978	16521	16072	15526	17066.5	15526	16522	17067.5	17067.5	16522	17067.5	17067.5	17067.5	16522	16522	17067	17067	17067	17067	17067	17067	-14433	17067	17067	17067

Fig. 2.6 The learning and erring process of Subject 1129

remains, even though he was very close to having a full grasp of the underlying environment and incessantly earned the highest profit from period 17 to the end (see the last row).

What is particularly interesting is the gambler's route as we have discussed in Sect. 2.3.1.1. Subject 1331 at a very early stage had already found the route for the risky higher profit. His learning index in periods 3, 4, and 6 of 2,096 or 2,098 shows his success in trading the second unit at a noncompetitive bid. This noncompetitive bid caused him to lose the deal in period 7, and he seemed to learn the risk associated with this lower bid with this loss, and decided to walk away from this gambler's route and never came back. This pattern clearly shows that when he recognized the risk and decided not to take the risk, period 7 was the critical point.

Figure 2.6 shows another example of discovering the gambler's route. Subject 1129 also found the gambler's route in period 13, when he tried to bid at an equal level to the bid of Buyer 4 with 6,987. He succeeded by "stealing" the deal and earned a higher profit. By the same greedy strategy, he also succeeded four times in the next five periods (periods 14–19). He did fail in period 15, but that single failure did not prevent him from walking on this gambler's route. Then the two consecutive losses in periods 20 and 21 finally made him realize that his luck was simply not as good as he expected. So, he also walked away from the gambler's route and never came back.

This pattern of "walking away from the gambler's route" is intriguing because human traders were not supplied with the information on the chance of getting the deal when the two bids were equal. They had to calculate the risk solely based on their experience. They may thus behave like the expected-profit maximizer except that, without being given the underlying probability, their profit expectations have to be formed through experience. The judgment of risk or risk preference is obviously different among traders. For Subject 1331, it took only one failure before he walked away, but, for Subject 1129, it took failures on three occasions.

Table 2.5 Learning and erring patterns represented by different plateaus

Plateau	Index	Points	Target earning	Description
A+	$X_3 + \alpha Y + \beta Z$	1,975–2,400	Higher	Lucky gambler
A	$X_1 + 4Y$	1,400	Y	Global optimizer
B	$X_1 + 4Y + \beta Z$	1,375–1,399	Y	First-order ε-global optimizer
C	$X_1 + 3Y + \beta Z$	1,275–1,300	Y	Second-order ε-global optimizer
D	$X_1 + 2Y + \beta Z$	1,175–1,200	Y	Third-order ε-global optimizer
	$X_1 + Y + \beta Z$	1,075–1,100		
	$X_1 + \beta Z$	975–1,000		
F	$X_2 + 3Y + \beta Z$	275–300	N	Non-optimizer
	$X_2 + 2Y + \beta Z$	175–200		
	$X_2 + 1Y + \beta Z$	75–100		
	$X_2 + \beta Z$	-25 to 0		

$\alpha \in \{1,2,3,4\}$, and $\beta \in \{0,1,2,\ldots,25\}$. Points given in the third column "points" are the results from the index formula (second column) using the parameters specified in Table 2.4

2.3.3 Plateaus

The credit assignment rule specified in Sect. 2.3.1 allows us to easily separate several different kinds of behavior, as is now summarized in Table 2.5, and which can be read as a sequence of plateaus arranged in descending order. Basically, we have classified traders into four distinct groups, namely, lucky gamblers (Class A+), optimizers (Class A), near optimizers (Classes B, C, and D), and non-optimizers (Class F).

Traders belonging to Class A+ have been described in Sect. 2.3.1.1. Traders belonging to the other four highest classes, Classes A, B, C, D, are those who are able to achieve target returns. However, what distinguishes Class A from other classes is that the traders belonging to the former are exactly on the trading schedule (market timing and bids), whereas the traders belonging to the latter are not. This can happen when the human trader misplaced an aggressive lower bid, which caused him to miss an early trade and fall out of the target trading schedule accordingly. Every such single miss can be interpreted as if human traders were still testing other possibilities of generating a higher profit by exploring around a small (ε) neighborhood of a global optimum. Hence, Classes B, C, and D can be pictured as traders who are in the small neighborhood of the global optimum with different radii, from a smaller one to a larger one. In the parlance of economics, if Class A is equivalent to rational traders, then Classes B, C, and D can be analogous to near-rational traders.

In contrast to the above four classes, Class F traders are traders who are still distant from the global optimum due to various errors (bidding, timing, etc.).

Despite the noticeable distance, some agents were able to figure out part of the structure of the trading game; they, therefore, made one, or two, or three deals in line with the trading schedule. However, since all of them are still in the early or the middle stage of learning, they are qualitatively separated from the global optimizers.

2.3.4 Learned or Not and When?

Based on the two illustrations in Sect. 2.3.2, can we use the learning index to decide whether the subject has actually learned the optimum strategy, and, if so, when? This issue is more subtle than what one might think. Using the examples above, can we consider a subject with a score of 1,400 to be the one who has learned? The answer is *yes*, if he could have repeatedly achieved this score, but what happens if he did not? The idea to be discussed below is to allow for a kind of deviation which we shall call an *accident* and to develop an *accident-tolerance criterion* for determining whether the agent has learned.

By that, we intend to consider the case where the subject seemed to learn the benchmark strategy, but his learning index was not consistently high as 1,400 and might occasionally have fallen down to a lower level (Fig. 2.5). These falls may occur for various reasons. First, the subject was tired, absent-minded, and made operational mistakes. Second, the subject was not sure that he had already found the benchmark strategy and attempted to explore further before realizing that nothing was there. Falls of these kinds can then be tolerated as long as they do not occur *frequently*. Hence, a subject is considered to have learned the benchmark strategy if he can stay on the high plateau long enough to make any fall look like an accident.

The discussion above motivates the development of the accident-tolerance criterion. What we propose is a $Q-q$ rule, where Q refers to the *length of window* denoting the most recent Q periods. Among the most recent Q periods, the subject is either on a high plateau or not: q_1 is the number of the periods that he stayed on a high plateau, and q_2 is the number of periods that he did not. Obviously, $Q = q_1 + q_2$. Now, consider the ratio $q = q_2/Q$. If the error is an accident, then q must be low enough to justify it being so. The question is how low. The answer may further depend on the subject's most recent Q location. Is it a global optimum or an epsilon-global optimum? Intuitively, q can be higher if the subject has already been in the global optimum, and lower if the subject has not. Quantitatively speaking, q should be an inverse function of ε (the radius of the neighborhood of the global optimum).

To implement this $Q-q$ rule, we have to parameterize it. What we suggest in this study is the following. We only consider Classes A and B (Table 2.5) as the high plateau, i.e., we take the first-order near-optimum as the threshold for the applicability of the $Q-q$ rule. Higher-order near-optima will make it difficult for us to distinguish accidental errors from true errors, which in turn will make it harder to catch the first crossing time that the subject learns the optimum. Other parameters are specified in Fig. 2.7 to satisfy a q as a monotone decreasing function of the radius of a neighborhood of the global optimum.

Criteria \ Period	−6	−5	−4	−3	−2	−1	Current period
LI = 1400					▨	▨	▨
1400 > LI ≥ 1395				▨	▨	▨	▨
1400 > LI ≥ 1390			▨	▨	▨	▨	▨
1400 > LI ≥ 1380		▨	▨	▨	▨	▨	▨
1400 > LI ≥ 1375	▨	▨	▨	▨	▨	▨	▨

Fig. 2.7 Accident-tolerance criteria for deciding whether the subject has learned

As suggested in the figure, it is sufficient to consider that the subject has learned the optimum strategy if he had the highest score (1,400) twice over the last three periods ($q = 1/3$). In other words, if he has been really good on two occasions, then missing once is accepted as an accident. In a similar vein, we also consider a subject to have learned if his score is between 1,395 and 1,400 three times over the last four periods ($q = 1/4$), or between 1,390 and 1,400 four times over the last five periods ($q = 1/5$), or between 1,380 and 1,400 five times over the last six periods ($q = 1/6$), or between 1,375 and 1,400 six times over the last seven periods ($q = 1/7$).

2.4 Heterogeneity in Learning

2.4.1 Time Required to Learn and the Aftermath

The learning index (Table 2.3) and the accident-tolerance criteria (Fig. 2.7) are now applied to the 165 subjects (Sect. 2.3.2.2). The results are shown in Fig. 2.8. To maintain brevity, we only show those subjects who have learned, at least once, in the sense of the accident-tolerance criteria. In other words, one of the five possibilities, as shown in Fig. 2.7, must apply for the subject at least once during the 30-period experiment; if that never happens, the subject simply did not learn the optimum and the code is not shown in this figure. In this way, the learning dynamics of 29 subjects (17.5% of the 165 subjects) are presented in Fig. 2.8. Their code is listed in the first column of the figure from the bottom to the top, based on the time used to learn the optimum; the lower, the faster.

Each row then denotes the state of each subject in each of the 30 periods of the experiment, from the left to the right. The blank cell means that the subject has not learned or learned but "forgot" in that respective period. The blue-colored cell means that one of the accident-tolerance criteria (Fig. 2.7) applies to the respective agent in the respective period. The first blue-colored cell in each row refers to the earliest time that the subject learned the optimum, and all other blue-colored cells following this leading one indicated that the subject stayed on the optimum strategy after having learned it.

As we can read from the bottom to the top, some subjects were able to learn the optimum strategy in the very early periods, like Subject 1531, who had already

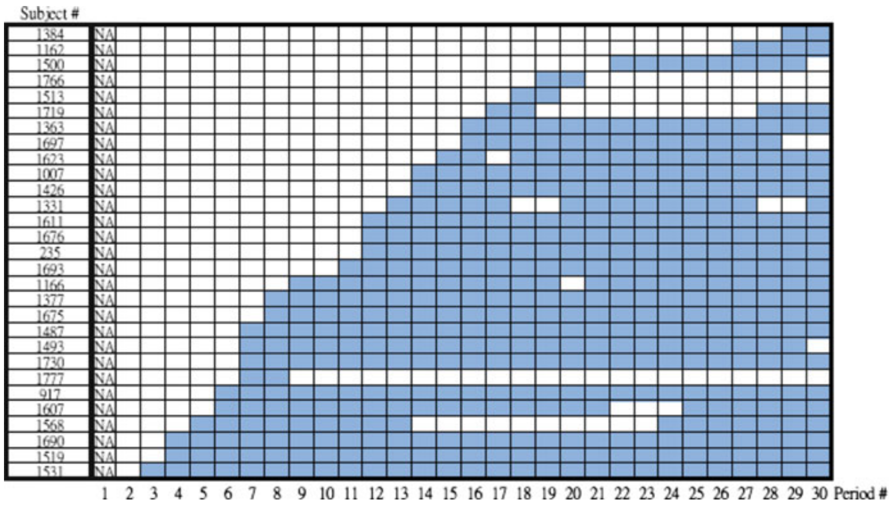


Fig. 2.8 Subjects who have learned the global optimal trading strategy: when it was the first time that the human traders learned the optimum trading strategy, and, after they learned it, whether they stayed on the optimal state. Out of 165 human traders, only 29 have learned the optimum before the expiration of the entire experiment. Their code is listed in the first column of the figure from the *bottom* to the *top*, based on the time used to learn the optimum; the lower, the faster. Each *row* then indicates the state of each subject in each of the 30 periods of the experiment, from the *left* to the *right*. A *blank cell* means that the subject has not learned or has learned but "forgot" during that respective period. The first *blue-colored* cell in each row indicates the earliest time that the subject learned the optimum, and all other *blue-colored* cells following this leading one indicate that the subject stayed on the optimum trading strategy after having learned it

done so in period 3; some needed a longer time to do so, like Subject 1384, who did not learn the optimum strategy until period 29. In addition to the minimum time required to learn the optimum, the aftermath of reaching the global optimum is also heterogeneous among agents. Once after being blue-colored, most subjects remain blue-colored, such as Subjects 1531, 1519, and 1690 (Fig. 2.8). There are a few, such as 1568 and 1607, who detoured from the optimum strategy to have further explorations, but were able to return in a later period. Only a very few, such as 1777, failed to come back again before the expiration of the experiment. Hence, in general, the global optimum is quite *stable* for most subjects: once they learn it, they will constantly keep it.

2.4.2 Cognitive Capacity

Our double auction experiment shows once again the heterogeneity of the learning dynamics among human traders. One of the most ambitious plans under the integration of agent-based computational economics and finance and experimental economics is to examine and model the great heterogeneity of human subjects as manifested in their learning dynamics.[12] There are two fundamental issues arising in this research direction. First, under the parsimony principle, how many attributes are needed for representing, up to a substantial degree, each human trader so that their heterogeneity in the learning dynamics can be replicated through artificial agents? Second, for each attribute and the assigned value, should we consider a different computational intelligence algorithm or the same algorithm but with different parameter values?

In this chapter, our focus is on the first issue and starts from a very fundamental level, namely, the psychological attribute of human traders.[13] By that, we mean differentiating human traders by either their *cognitive attributes* or *personality attributes*. The key measurement of the former is *working memory capacity*, whereas the key measurement of the latter is a Chinese version of *Big Five*. In this section, we shall focus on the former and provide details of the latter in a separate section (Sect. 2.4.3).

2.4.2.1 Working Memory Capacity

Cognitive capacity is a general concept used in psychology to describe a human's cognitive flexibility, verbal learning capacity, learning strategies, intellectual ability, etc. [13]. Although cognitive capacity is a very general concept and can be measured from different aspects with different tests, concrete concepts such as the intelligence quotient (IQ) and working memory capacity are considered to be highly representative of this notion. We adopt working memory capacity as a measure of cognitive capacity because working memory capacity is not simply a measurement of the capacity of short-term memory, but a "conceptual ragbag for everything that is needed for successful reasoning, decision making, and action planning" (p. 167, [47]). It has been shown that WMC is highly correlated with general intelligence [26, 39] and performance in other cognitive domains, such as sentence comprehension [27] and reasoning [41]. Recently, working memory capacity has been regarded as an important economic variable in both experimental economics and agent-based computational economics.[14]

In this study, human traders involved in the double auction experiments are requested to take a working memory test. The test version is based on [42] and is composed of five parts, which are backward digit span (BDG), memory updating (MU), operation span (OS), sentence span (SS), and spatial short-term

[12]For those readers who are unfamiliar with this development, some backgrounds are available from [19, 20].

[13]While the conversation between psychology and economics has a long history and a rapidly growing literature, it was only very recently that economists started to take into account psychological attributes in their economic modeling and analysis.

[14]A survey is available from [21].

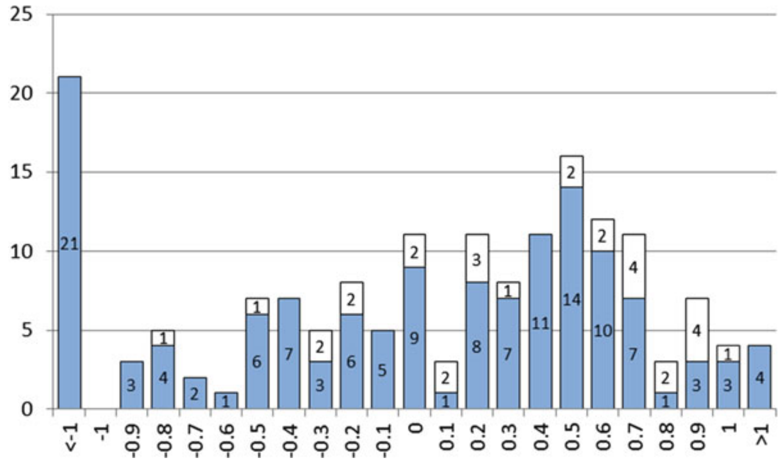


Fig. 2.9 Score distribution of the working memory test

memory (SSTM). They basically ask the subjects to undertake some tasks, such as memorizing series of numbers and letters and performing very basic arithmetical operations. By following the conventional procedure in psychological tests, the scores for each task are normalized using the mean and the standard deviation of the subject pool. The five standardized scores for the five tasks will then be averaged to arrive at the WMC of a specific subject. The histogram of the normalized WMC is depicted in Fig. 2.9. The histogram starts with the leftmost cluster "WMC < -1" (21 subjects in this cluster) and ends at the rightmost cluster "WMC > 1" (4 subjects). In between, there are 20 clusters, each with a range of 0.1, dividing the distribution equally into 20 equal intervals, from $[-1, -0.9)$, $[-0.9, 0.8)$, ..., all the way up to $[0.9, 1]$.

2.4.2.2 Optimum-Discovery Capability

To have a general picture of how WMC may actually impact the capability to discover the pattern or to learn the optimum strategy, we also indicate, within each WMC cluster, the number of human traders who were able to discover and learn the optimum and place it on a higher layer (blank-colored) to be separated from the number of those who did not on a lower layer (blue-colored).

By just eye-browsing Fig. 2.9, we can see that most human traders who were able to discover the pattern have a positive WMC (23 out of 29), and for those subjects who have a WMC below -0.5, only one, out of 32, is able to do so. Therefore, there is evidence indicating a positive influence of WMC on the pattern discovering capability. To present the result in a more precise manner, we also average the WMC of the performing group (29 subjects) and compare it with that of the nonperforming

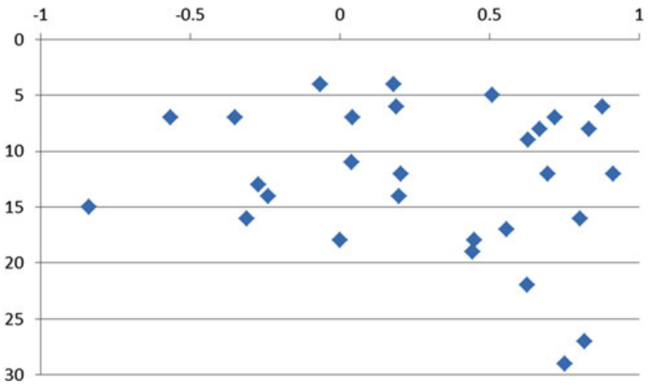


Fig. 2.10 First time and WMC

group (136 subjects). It is found that the mean WMC is 0.28036 for the performing group, but only −0.12648 for the nonperforming group; the whole population average is −0.05004.

Despite the positive statistics between WMC and the optimum-discovery capability, it is also interesting to notice the existence of some "outliers." Specifically, the four subjects belonging to the highest cluster of WMC (>1) and the three out of four in the next highest cluster ([0.9,1]) all failed to discover the global optimum. We shall come back to this point later in Sect. 2.4.2.4.

2.4.2.3 Time Required to Discover

To have a further look at the effect of cognitive capacity, we examine, within the performing group, whether the trader with a *higher* WMC tends to discover or learn the optimum *faster*. To do so, Fig. 2.10 gives the $X - Y$ plot of the pairs between WMC and the discovering time for each trader belonging to the performing group. Subject 1531, the trader who used the minimal time to discover the optimum, spent only three periods to "touch down" and stay on the optimum strategy almost for the entire duration (Fig. 2.8); yet, his WMC is only in the middle, 0.2475, and not particularly high.[15] In fact, the whole $X - Y$ plot of Fig. 2.10 shows that there is no significant relationship between WMC and discovering time, except for the following interesting finding. After 15 periods, most low WMC traders were no longer able to learn the optimum; more working time for them is to no avail.

[15]From Fig. 2.7, for a trader who is identified as a case of learning the optimum in period 3, his learning index must be in plateau A in the first three periods. Actually, Subject 1531 started performing the optimal strategy in period 2 until the end of the experiment except for one period obviously due to a typo. Subject 1531 seems to thoroughly understand the market features in period 1, and then performs the optimum strategy seamlessly in period 2.

Table 2.6 Working memory capacity and staying frequencies in the high and low plateaus

Class (plateau)	Frequencies	Cardinality	Mean WMC
A–D	15	38	0.31396
	20	30	0.33392
	25	18	0.24462
F	30	121	−0.17702
	10	139	−0.11911

However, this "bottleneck" does not exist for the high WMC traders. In fact, the nine subjects who discovered the optimum after period 15 all have positive WMC, except one.

The bottleneck observed in Fig. 2.10 can be regarded as a cognitive trap. The depth of the trap might be different for traders with different WMC. For traders with lower WMC, this trap might be too deep to jump out; hence, if these traders do not initially stand in a favorable position outside the trap, a longer learning time may help them a little. That is why we see that very few can walk out of this trap in the entire second half of the experiment. The attempt to associate the *energy* required to climb the hill with *cognitive capacity* is first made in [24]. Here, we also have some observations similar to this psychological analogy of the numerical trap.[16]

2.4.2.4 Target Return

The purpose of this section is to see whether cognitive capacity may have effects on the learning behavior of human traders. To do this, we separate human traders into different groups, for example, those who learned the optimum and those who did not, and then examine whether the WMCs of these two groups are different. We can also consider other grouping possibilities. One of them is to differentiate subjects by the frequencies according to which they stayed in different plateaus. Hence, we can differentiate those frequently visiting a high plateau from those frequently visiting a low plateau. For both, we may further differentiate them by *degree*, such as normal frequency or high frequency. We exemplify this kind of grouping in Table 2.6.

[16] This chapter and [24] are both under the umbrella of a 3-year NSC research project. Hence, they both share some similar features. What distinguishes [24] from this chapter is that the former explicitly constructs traders' learning paths in a numerical landscape. The question is then to address whether the observed learning behavior of traders can also be understood as an output of a numerical search algorithm. In other words, they inquire whether there is a connection between *behavioral search* and *numerical search*. However, the trading environment here makes it hard to derive this geometrical representation; therefore, the use of a learning index becomes another way to see how this trap might actually also exist. Despite this difference, the implication of these two studies is the same: *we need to equip artificial agents with different CI tools so that their search behavior can be meaningfully connected to the cognitive capacity of human traders, or, more directly, we need to reflect upon the cognitive capacity of different CI tools* [19].

The upper panel of Table 2.6 denotes the group of traders who are frequently classified into the high plateau, namely, Classes A, B, C, and D. In other words, they are the traders who frequently meet the target return. By different frequencies, we further consider three types of traders: those who were classified into this high plateau at least 15 times, at least 20 times, and at least 25 times, abbreviated as "15," "20," and "25" in the second column of the table. Similarly, the lower panel refers to the traders who are frequently classified into the low plateau (Class F) and who failed to earn the target return. The two subgroups "30" and "10" refer to the types of traders who were classified into this low plateau all the time (30 times) and at least 10 times, respectively.

These five groups are not exclusive to each other. In fact, group "25" is obviously a subset of group "20," which in turn is also a subset of group "15." As we can see from the 3rd column of Table 2.6, there are 38 subjects belonging to group "15," but only 30 of them belong to group "20" and 18 belong to group "25." Similarly, group "30" is also a subset of group "10." There are 139 subjects belonging to the former, whereas only 121 out of these 139 belong to the latter. These five groups provide us with another opportunity to see the effect of cognitive capacity on trading performance.

First of all, very similar to our earlier analysis of the optimum-discovering capability, we find that traders frequently classified to F tend to have a lower WMC than those who were frequently classified to a high plateau. As one can see from the fourth column of Table 2.6, the former has an average of negative normalized WMC, whereas the latter has a positive average. Both the student t-test and Wilcoxon rank sum test show that the difference in WMC between these two classes, "F" and "A–D", is significant. Second, if we further isolate group "30" (those who failed to make the target return from the beginning to the end), then it has a particularly lower WMC (-0.17702), which is lower than that of group "10" (-0.11911). Third, however, if we do the same thing for the high plateau and isolated group "25," we shall be surprised by the result that this elitist group (traders who can make the target return 25 out of 30 times) does not have a higher WMC (0.24462) than its super sets, "15" (0.31396) and "20" (0.33392). This later evidence is very intriguing, because this finding accompanied by our early observations of the outliers in the "optimum-discovery" section (Sect. 2.4.2.2) together lend support to the hypothesis of *a diminishing marginal contribution of cognitive capacity* in the psychological literature.[17]

[17]As the psychological literature points out, high intelligence does not always contribute to high performance—the significance of intelligence in performance is more salient when the problems are more complex [25]. In addition, it appears that intelligence exhibits a decreasing marginal contribution in terms of performances [29, 37]. In the setting of an agent-based double auction market, Chen et al. [23] have replicated this diminishing marginal contribution of cognitive capacity. In that article, autonomous traders are modeled by genetic programming with different population size. The population size is manipulated as a proxy variable for working memory capacity. They then found that, while the trading performance between agents with small

2.4.3 Personality Traits

The second possible contributing factor for the observed heterogeneity among human traders in their learning behavior is personality. In personality psychology, there are many competing paradigms, but the personality trait is now the most widely accepted theory. Although there have been many studies regarding traits, they never seem to agree on the number of basic traits. Nevertheless, the *five-factor model* developed by Gordon Allport (1897–1967) is probably the most popular one. The five personality factors, also known as *Big Five*, are *openness*, *conscientiousness*, *extraversion*, *agreeableness*, and *neuroticism*. A convenient acronym is "OCEAN." Using the Big Five model, labor economists have already started to explore the relevance of personality to economics [14]. Big Five has also been applied to economic experiments [9–11, 35], while a lot of other measures are simultaneously used. Recently, there is even a trend to combine both cognitive and noncognitive factors to account for economic behavior [1, 24].

In this study, we applied a variant of Big Five to measure the personality trait of human traders, which we shall call Big Seven. The Big Seven was originally developed by Kuo-Shu Yang, a Fellow of Academia Sinica. Yang considered that Big Five may not be straightforwardly applied to Chinese due to their different cultural background; he, therefore, revised the Big Five model and made it the Big Seven model. The seven factors are *smartness*, *conscientiousness*, *agreeableness*, *trustworthiness*, *extraversion*, *chivalry*, and *optimism*.[18] A test accompanying this Big Seven model is also developed. The test is organized into seven major categories. For each category, there are 15–20 short descriptions (adjectives) of personality. As a total, there are 131 descriptions. For each description, the subject is required to rate himself on a scale from 1 (unsuitable) to 6 (suitable). Hence, "1," if he thinks that the adjective does not at all describe their personality, and "6," if he thinks that the adjective perfectly describes his personality. After filling in all 131 entries, his personality score will be calculated based on the loading (weight) of each entry.

The basic personality statistics of all 168 subjects are summarized in Table 2.7. The first two rows give the number of queries (each query is associated with an adjective) and the range of score. The third, fourth, and fifth rows give the mean of these scores over the whole group of 168 subjects, over the performing subgroup (those who were able to discover or learn the optimum) and over the non-performing subgroup (those who did not), respectively.

---

population size and agents with a large one is significantly different, this difference between agents with a *large* one and agents with a *larger* one is negligible.

[18]There is no official translation of the seven factors. An attempt to do so on our own is not easy, in particular if one wants to describe the whole of 15–20 adjectives using a single word, such as conscientiousness. What we do here is to follow OCEAN closely and to use the same name if the factor in Big Seven shares very much in common with one of the Big Five. Examples are conscientiousness, agreeableness, and extraversion.

Table 2.7 Big-7 personality and discovery of the optimum

	S	C	A	T	E	Ch	O
# of queries	20	20	20	20	20	20	15
Range of score	20−120	20−120	20−120	20−120	20−120	20−120	15−90
Avg. (whole)	84.58	71.52	79.50	86.39	76.79	68.11	54.83
Avg. (performing)	86.43	71.90	78.93	90.78	79.00	70.56	56.87
Avg. (nonperforming)	84.15	71.43	79.63	85.36	76.27	67.54	54.35
Difference	2.28	0.47	−0.69	5.42	2.72	3.01	2.52

The first row is the initial of each of the Big Seven: smartness (S), conscientiousness (C), agreeableness (A), trustworthiness (T), extraversion (E), chivalry (Ch), and optimism (O)

Our analysis starts with a quick look at how the performing group and non-performing group differ in their Big Seven. For this purpose, the sixth column of Table 2.7 gives the differences between the two groups in these seven items. By looking at their relative magnitudes, among the seven, trustworthiness stands out, as the most salient one to distinguish the two groups, followed by chivalry, extraversion, optimism, and smartness. The two with rather small magnitudes are agreeableness and conscientiousness. These results are not well expected. First of all, conscientiousness, which has been constantly identified as a factor to predict economic behavior, is not founded here,[19] while our finding is consistent with this literature by singling out the extraversion as one of the top three. Second, the most salient one found in our dataset is trustworthiness.[20] This is the one which has never been mentioned in a separate study [24], which also examines the role of Big Seven in traders' performance.[21]

Like what we do for cognitive capacity (Fig. 2.9), we also plot the histogram of the Big-Seven personality score in Fig. 2.11, and, at the top of each bar, we indicate the number of the human traders who were able to discover the optimum. To make it easier to see the relationship between the seven factors and the capability to discover, we further rescale the presentation in terms of percentages. Hence, each cluster of each factor has a total of 100%, which is then divided by the size (share) of the blank-colored area (the performing one) and the blue-colored area (the nonperforming one). In this way, we can easily compare the relative size across different clusters.

By eye-browsing the seven histograms, one may find it difficult to see any visualizable pattern between personality factors and optimum-discovering capability except for the factor trustworthiness. In the case of trustworthiness, one can

[19]Conscientiousness is found to be a good predictor of job performance, mortality, divorce, educational attainment, car accidents, and credit score [1, 7, 8, 24, 38, 49].

[20]In fact, both our student t-test and Wilcoxon rank sum test only find this factor statistically significantly different between the performing group and the non-performing group.

[21]However, this study differs from [24] in using a different performance measure. See also footnote 16.

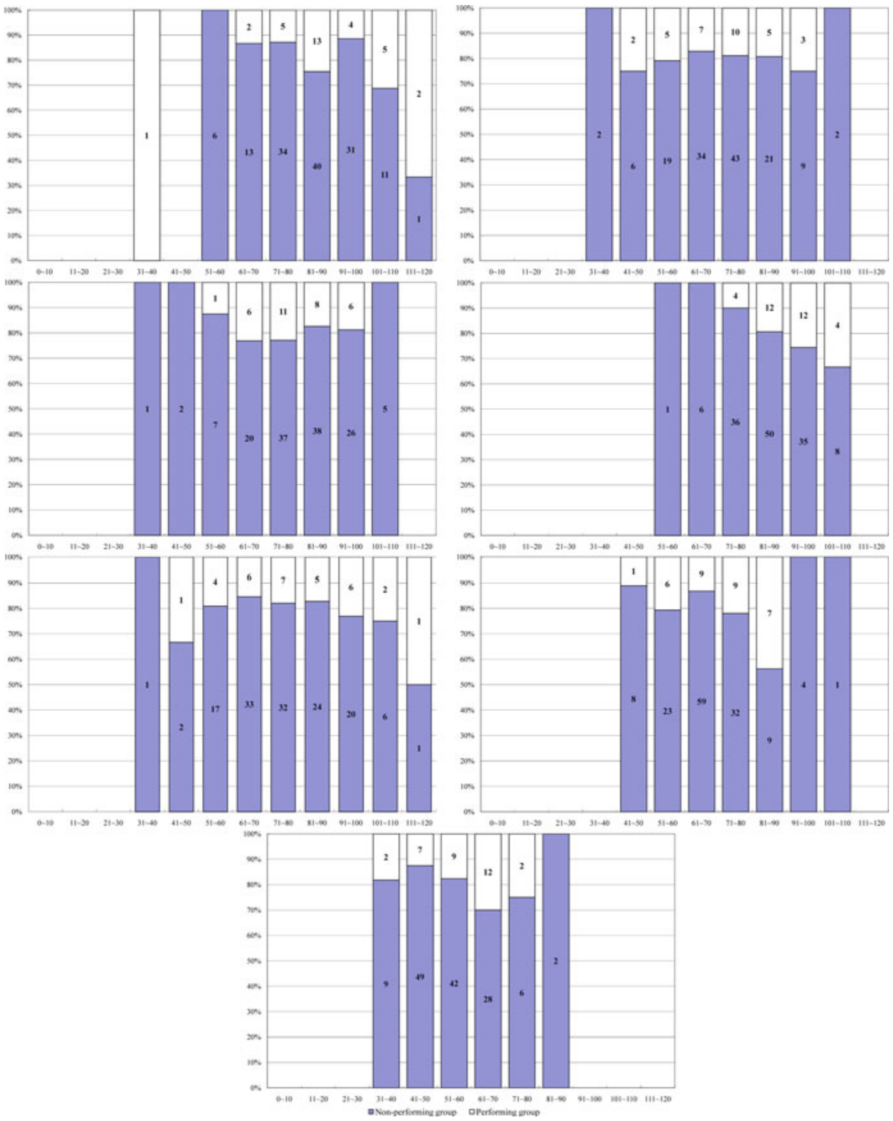


Fig. 2.11 Big-7 and learning performance: the seven figures, from the first row to the fourth row and from the *left* to the *right*, refer to the personality factor, smartness (*1st row, left*), conscientiousness (*1st row, right*), agreeableness (*2nd row, left*), trustworthiness (*2nd row, right*), extraversion (*3rd row, left*), chivalry (*3rd row, right*), and optimism (*4th row*), respectively

see that the percentage of performing human traders increases with the score of trustworthiness. This finding is consistent with the one revealed in Table 2.7. However, it is hard to argue why trustworthiness may help discover the optimum and it becomes even harder when other seemingly natural ones, such as conscientiousness, extraversion, and even smartness, all fail to play a role.[22] This may in fact indicate the potential problem in this self-reported evaluation.

Let us use Subject 917 to illustrate this problem. This smartness factor of this subject is self-reportedly between 30 and 40, i.e., the lowest one of the whole sample of 168 subjects. Nevertheless, he is one of the 32 subjects who were able to discover the optimum, and, therefore, behaves as an "outlier" in the sense of the only performing trader with a very low score in smartness. On the other hand, his WMC score is 0.8756, which is among the top 10%. So, in contrast to his WMC score, his self-reported smartness seems to be too much understated.

2.5 Concluding Remarks

The chapter, distinguished from most financial applications of computational intelligence in economics, is concerned with the issue of how computational intelligence can help build artificial traders who are capable of mimicking human traders' learning and erring behavior [18]. This research area is just in its very beginning stage, but it is important in the following two ways.

Firstly, there is no clear evidence indicating that human traders (human heuristics) can be or have been substantially replaced by robots (machine heuristics). While both cognitive psychology and computational intelligence have *heuristics* as their research interest, they seem to have been developed at different levels, in different directions, and have been applied to different domains. When coming to competition, what often surprises us is that, even in a simple situation like the double auction market, there is no clear evidence that heuristics developed by machines can outperform those coming out of humans.

Take our double auction experiments as an example. It is highly interesting to see how human traders actually learned, consciously or unconsciously, the *two-stage* procrastination strategy, particularly when a large number of runs of GP could only find the *one-stage* procrastination strategy. The one-stage procrastination has a simple heuristic behind it: hold everything till the end of a trade. This simplicity can be easily picked up by human traders with good intuition. Hence, one may

[22]Among the seven factors, Chen et al. [24] find conscientiousness, extraversion, and agreeableness to be influential, at least in some contexts. In their analysis, they attempt to justify each of these three. Among the three, conscientiousness is probably the easiest one to justify, given the already lengthily archived documents (see footnote 2.4.3). They then go further to justify the other two by using [28] to argue that extraverted subjects are more sensitive to potential rewarding stimuli through the mesolimbic dopamine, which may in turn help them more easily find the more profitable trading arrangements. In addition, for agreeableness, they argue that subjects with a higher degree of agreeableness can resist time pressure and may be able to think for a longer time before making decisions.

say that GP simply replicates human traders' heuristics or "rules of thumb." Hence, the heuristic developed by GP is consistent with the heuristics developed by many human traders.

Nonetheless, the two-stage procrastination bidding is less straightforward. GP succeeded in learning this strategy in only a few runs; most runs were trapped in a local optimum. However, as we can see from our human-trader experiments, some human traders could actually learn this strategy, while most of them also failed. Among the successful ones, some could even touch down in a few minutes. Evidently, for them, they did not learn this by solving a hard combinatoric optimization; instead, they learned it as a heuristic.

However, why were some people able to see this pattern, whereas others failed to do so? If we assume that some have a good representation of the problem, by that representation the problem becomes easier, but others do not "see" this pattern. Then, is this difference mainly due to chances, which are very much random, or can this difference be attributed to some more fundamental causes?

To answer this question, this chapter proposes a new research framework. The new research framework includes the design of a novel learning index in light of a separation hyperplane. Through this index, one can better capture the state of human traders' learning dynamics and then develop various performance measures upon it. This performance measure, which encapsulates human traders' heterogeneity in learning dynamics, can be further analyzed in light of the basic attributes of human traders, such as their cognitive capacity and personality traits. This chapter, to the best of our knowledge, is one of the pioneering studies devoted to the inquires about whether cognitive capacity and personality have influences on traders' performance and their heuristic development in a laboratory setting.

Very similar to the earlier study [24], we once again confirm the significance of cognitive capacity to the heuristic development of human traders. However, we do not find a particularly insightful connection between personality and learning dynamics; in particular, the factor conscientiousness plays no role in the heuristic formation, a result different from that of the earlier study.

Secondly, tremendous analyses of heuristic biases of human traders are conducted in the area of psychology, cognitive neuroscience, and behavioral finance, but this literature, at this point, receives little interdisciplinary collaboration with the computational intelligence society. We believe that narrowing this gap may help us build more human-like artificial traders. Hence, carefully studying human heuristics with computational intelligence may lead us to explore a rich class of fast and frugal heuristics [32].

An emotional and neurocognitive study of financial decision making may benefit from the analysis provided in this chapter. It enables us to stand in the front line to see interesting behavior patterns which can be further explored from the aspects of psychological analysis, and also enables us too see how such advanced analysis can enrich the current literature on behavioral economics.

Acknowledgements The authors are grateful to Professor Kuo-Shu Yang for his generous permission for using his developed Chinese version of the Big-Five personality test. We are also grateful to Professor Li-Jen Weng and Professor Lei-Xieng Yang for their advice and guidance on the psychological tests implemented in this study. NSC research grants no. 98-2410-H-004-045-MY3, no. 99-2811-H-004-014, and no. 100-2410-H-029-001 are also gratefully acknowledged.

Appendix: Double Auction as a Combinatorial Optimization Problem

In order to set the benchmark, we have to determine the best bidding strategy for Buyer 1. However, to determine the best timing and bidding values in a discrete-time double auction is a combinatorial optimization problem and is NP-hard. To tackle this problem, one should first notice that the discrete-time double auction is in fact an integer programming problem constrained by various rules and market regulations, such as that a buyer should always bid from his/her highest token to the lowest token and he/she can only bid once in each step. Second, to solve this integer programming problem, Xia et al. [56] have shown the superiority of the branch-and-bound method. We therefore use the branch-and-bound method to find the optimal bidding strategy for Buyer 1 in our markets. We will first demonstrate the rules of bidding and matching, and the exact models of integer programming will then be given as well.

In the SFTE auction, there are nb buyers and ns sellers in the market. The bidding/asking rules as well as the transaction mechanism are as follows:

1. Each trader has nt ordered tokens. Starting from the first token, traders have to bid/ask based on the ordered token values.
2. The auction lasts np steps. In each step, every buyer and seller can choose to bid/ask or remain silent.
3. Only those who bid/ask in the market have chances to make transactions, and only those who make transactions earn profits. Staying silent will not pay.
4. The bidder with the highest bid price in a step is the current buyer, while the seller with the lowest ask price in that step is the current seller.
5. Only the current buyer and the current seller have the chance to complete a transaction, depending on whether the bid price is higher than the ask price.
6. The transaction price is the average of the current buyer's bid price and the current seller's ask price.
7. Once a trader is involved in a transaction, he/she naturally uses up one token and has to continue trading based on the next token value.
8. If a trader uses up all nt tokens, he/she is expelled from the market and can only return to the market when the market starts over again.
9. In our human experiments as well as GP simulations, all traders except human subjects or GP traders are truth tellers, who always truthfully use their token values as the bids or asks and will never keep silent in the market.

Table 2.8 Parameters and variables

b	Index for buyers
s	Index for sellers
j	Index for steps
k	Index for tokens
$btv_{b,k}$	Value of token k for buyer b
$stv_{s,k}$	Value of token k for seller s
bm	A very big number
nb	Number of buyers
ns	Number of sellers
np	Number of steps
nt	Number of tokens
$BT_{b,k,j}$	Whether buyer b's token k is bidded in the market in step j [0,1]
$AT_{s,k,j}$	Whether seller s's token k is asked in the market in step j [0,1]
$B_{b,j}$	Buyer b's bid in step j
$A_{s,j}$	Seller s's ask in step j
$CB_{b,j}$	Whether buyer b's bid is the highest bid in step j [0,1]
$CS_{s,j}$	Whether seller s's ask is the lowest ask in step j [0,1]
$\overline{B}_j$	The highest bid in step j
$\underline{A}_j$	The lowest ask in step j
$T^B_{b,k,j}$	Whether buyer b's token k reaches a transaction in step j [0,1]
$T^S_{s,k,j}$	Whether seller s's token k reaches a transaction in step j [0,1]
T_j	Whether a transaction is made in step j [0,1]
P_j	The transaction price in step j

Our goal is to find the best bidding strategy, which maximizes Buyer 1's profit in the face of truth telling opponents. Since this is an integer programming problem, we can describe the problem with the objective function (2.1) and constraints ((2.2)–(2.21)). Notice that the constraints are proposed here to directly or indirectly enforce the rules and mechanisms developed above. Notations used in the objective function and the constraints are summarized in Table 2.8.

Objective Function

$$\max \sum_{j}^{np}\sum_{k}^{nt}[(btv_{1,k}-P_j)\times T^B_{1,k,j}] \tag{2.1}$$

Constraint for Rule 1

$$\forall b,\ BT_{b,1,1}=1\ \ \|\ \ \forall s,\ AT_{s,1,1}=1 \tag{2.2}$$

Constraint for Rule 2

$$\forall b,j,\ \sum_{k}^{nt}BT_{b,k,j}\le 1\ \ \|\ \ \forall s,j,\ \sum_{k}^{nt}AT_{s,k,j}\le 1 \tag{2.3}$$

Constraints for Rule 3

$$\forall b,j,k,\ T^B_{b,k,j} \leq BT_{b,k,j}\ \ \|\ \ \forall s,j,k,\ T^S_{s,k,j} \leq AT_{s,k,j} \tag{2.4}$$

$$\forall b,j,\ CB_{b,j} \geq \sum_k^{nt} T^B_{b,k,j}\ \ \|\ \ \forall s,j,\ CS_j \geq \sum_k^{nt} T^S_{s,k,j} \tag{2.5}$$

Constraints for Rule 4

$$\forall j,\ \overline{B}_j = \sum_b^{nb}(CB_{b,j} \times B_{b,j})\ \ \|\ \ \forall j,\ \underline{A}_j = \sum_s^{ns}(CS_{s,j} \times A_{s,j}) \tag{2.6}$$

$$\forall b,j,\ \overline{B}_j \geq B_{b,j}\ \ \|\ \ \forall s,j,\ \underline{A}_j \leq A_{s,j} \tag{2.7}$$

$$\forall j,\ \sum_b^{nb} CB_{b,j} = 1\ \ \|\ \ \forall j,\ \sum_s^{ns} CS_{s,j} = 1 \tag{2.8}$$

Constraints for Rule 5

$$\forall j,\ \sum_b^{nb}\sum_k^{nt} T^B_{b,k,j} \leq 1\ \ \|\ \ \forall j,\ \sum_s^{ns}\sum_k^{nt} T^S_{s,k,j} \leq 1 \tag{2.9}$$

$$\forall j,\ \sum_b^{nb}\sum_k^{nt} T^B_{b,k,j} = \sum_s^{ns}\sum_k^{nt} T^S_{s,k,j} \tag{2.10}$$

$$\forall j,\ (\overline{B}_j - \underline{A}_j) \times T_j \geq 0 \tag{2.11}$$

$$\forall j,\ \sum_b^{nb}\sum_k^{nt} T^B_{b,k,j} = T_j \tag{2.12}$$

Constraints for Rule 6

$$\forall j,\ (\overline{B}_j - \underline{A}_j) < T_j \times bm \tag{2.13}$$

$$\forall j,\ 2 \times P_j = (\overline{B}_j + \underline{A}_j) \tag{2.14}$$

Constraints for Rule 7

$$\forall b,k,\ \sum_j^{np} T^B_{b,k,j} \leq 1\ \ \|\ \ \forall s,k,\ \sum_j^{np} T^S_{s,k,j} \leq 1 \tag{2.15}$$

$$\forall b, j \in 2 \ldots np,\ BT_{b,1,j} = BT_{b,1,j-1} - T^{B}_{b,1,j-1} \ \|$$
$$\forall s, j \in 2 \ldots np,\ AT_{s,1,j} = AT_{s,1,j-1} - T^{S}_{s,1,j-1} \tag{2.16}$$
$$\forall b, k \in 2 \ldots nt, j \in 2 \ldots np,\ BT_{b,k,j} = BT_{s,k,j-1} + T^{B}_{s,k-1,j-1} \ \|$$
$$\forall s, k \in 2 \ldots nt, j \in 2 \ldots np,\ AT_{s,k,j} = AT_{s,k,j-1} + T^{S}_{s,k-1,j-1} \tag{2.17}$$

Constraints for Rule 8

$$\forall b,\ \sum_{j}^{np}\sum_{k}^{nt} T^{B}_{b,k,j} \leq nt \ \| \ \forall s,\ \sum_{j}^{np}\sum_{k}^{nt} T^{S}_{s,k,j} \leq nt \tag{2.18}$$

$$\forall s,\ \sum_{j}^{np} T^{S}_{s,nt+1,j} = 0 \tag{2.19}$$

Constraints for Rule 9

$$\forall b \in 2 \ldots 4, j,\ B_{b,j} = \sum_{k}^{nt}(BT_{b,k,j} \times btv_{b,k}) \tag{2.20}$$

$$\forall s, j,\ A_{s,j} = \sum_{k}^{nt}(AT_{s,k,j} \times stv_{s,k}) \tag{2.21}$$

References

1. J. Anderson, S. Burks, C. DeYoung, A. Rustichinid, *Toward the Integration of Personality Theory and Decision Theory in the Explanation of Economic Behavior* (University of Minnesota, Mimeo, 2011)
2. J. Arifovic, Genetic algorithms learning and the cobweb model. J. Econ. Dyn. Contr. **18**(1), 3–28 (1994)
3. J. Arifovic, R. McKelvey, S. Pevnitskaya, An initial implementation of the Turing tournament to learning in repeated two-person games. Game. Econ. Behav. **57**(1), 93–122 (2006)
4. B. Arthur, On designing economic agents that behave like human agents. J. Evol. Econ. **3**(1), 1–22 (1993)
5. H. Baker, J. Nofsinger, *Behavioral Finance: Investors, Corporations, and Markets* (Wiley, New York, 2010)
6. B. Barber, T. Odean, Trading is hazardous to your wealth: The common stock investment performance of individual investors. J. Finance **55**(2), 773–806 (2000)
7. M. Barrick, M. Mount, The big five personality dimensions and job performance: A meta-analysis. Person. Psychol. **44**(1), 1–26 (1991)
8. M. Barrick, M. Mount, T. Judge, Personality and performance at the beginning of the new millennium: What do we know and where do we go next? Int. J. Sel. Assess. **9**, 9–30 (2001)

9. A. Ben-Ner, F. Halldorsson, Measuring Trust: Which Measures Can be Trusted? Working paper, University of Minnesota (2007)
10. A. Ben-Ner, L. Putterman, F. Kong, D. Magan, Reciprocity in a two-part dictator game. J. Econ. Behav. Organ. **53**(3), 333–352 (2004)
11. A. Ben-Ner, F. Kong, L. Putterman, Share and share alike? Gender-paring, personality, and cognitive ability as determinants of giving. J. Econ. Psychol. **25**, 581–589 (2004)
12. Y. Bereby-Meyer, A. Roth, The speed of learning in noisy games: Partial reinforcement and the sustainability of cooperation. Am. Econ. Rev. **96**(4), 1029–1042 (2006)
13. Z. Boender, J. Ultee, S. Hovius, Cognitive capacity: No association with recovery of sensibility by Semmes Weinstein test score after peripheral nerve injury of the forearm. J. Plast. Reconstr. Aesthetic Surg. **63**(2), 354–359 (2010)
14. L. Borghans, A. Duckworth, J. Heckman, B. Weel, The Economics and Psychology of Personality Traits, NBER Working Paper No. 13810 (2008)
15. P. Bossaerts, What decision neuroscience teaches us about financial decision making. Annu. Rev. Financ. Econ. **1**, 383–404 (2009)
16. S.-H. Chen, Computational intelligence in agent-based computational economics, in *Computational Intelligence: A Compendium*, ed. by J. Fulcher, L. Jain (Springer, New York, 2008), pp. 517–594
17. S.-H. Chen, Genetic programming and agent-based computational economics: from autonomous agents to product innovation, in *Agent-Based Approaches in Economic and Social Complex Systems*, ed. by T. Terano, H. Kita, S. Takahashi, H. Deguchi (Springer, New York, 2008), pp. 3–14
18. S.-H. Chen, Software-agent designs in economics: An interdisciplinary framework. IEEE Comput. Intell. Mag. **3**(4), 18–22 (2008)
19. S.-H. Chen, Collaborative computational intelligence in economics, *in Computational Intelligence: Collaboration, Fusion and Emergence*, ed. by C. Mumford, L. Jain (Springer, New York, 2009), pp. 233–273
20. S.-H. Chen, Y.-L. Hsieh, Reinforcement learning in experimental asset markets. E. Econ. J. **37**(1), 109–133 (2011)
21. S.-H. Chen, S. Wang, Emergent complexity in agent-based computational economics. J. Econ. Surv. **25**(3), 527–546 (2011)
22. S.-H. Chen, R.-J. Zeng, T. Yu, Co-evolving trading strategies to analyze bounded rationality in double auction markets, in *Genetic Programming Theory and Practice VI*, ed. by R. Riolo (Springer, New York, 2008), pp. 195–213
23. S.-H. Chen, C.-C. Tai, S. Wang, Does cognitive capacity matter when learning using genetic programming in double auction markets? in *Multi-Agent-Based Simulation*, ed. by G. Di Tosto, H. Van Dyke Parunak. Lecture Notes in Artificial Intelligence, vol. 5683 (Springer, New York, 2010), pp. 37–48
24. S.-H. Chen, U. Gostoli, C.-C. Tai, K.-C. Shih, To whom and where the hill becomes difficult to climb: Effects of personality and cognitive capacity in experimental DA markets. Adv. Behav. Finance Econ (forthcoming)
25. R. Christal, W. Tirre, P. Kyllonen, Two for the money: speed and level scores from a computerized vocabulary test, in *Proceedings of the Ninth Annual Symposium, Psychology in the Department of Defense (USAFA TR '84-2)*, ed. by G. Lee, Ulrich (U.S. Air Force Academy, Colorado Springs, 1984)
26. A. Conway, M. Kane, R. Engle. Working memory capacity and its relation to general intelligence. Trends Cognit. Sci. **7**(12), 547–552 (2003)
27. M. Daneman, P. Carpenter, Individual differences in integrating information between and within sentences. J. Exp. Psychol. Learn. Mem. Cognit. **9**, 561–584 (1983)
28. R. Depue, P. Collins, Neurobiology of the structure of personality: Dopamine, facilitation of incentive motivation, and extraversion. Behav. Brain Sci. **22**(3), 491–517 (1999)

29. D. Detterman, M. Daniel, Correlations of mental tests with each other and with cognitive variables are highest for low-IQ groups. Intelligence **13**, 349–359 (1989)
30. M. Fenton-O'Creevy, E. Soane, N. Nicholson, P. Willman, Thinking, feeling and deciding: The influence of emotions on the decision making and performance of traders. J. Organ. Behav. **32**(8), 1044–1061 (2011)
31. M. Gendreau, J.-Y. Potvin, *Handbook of Metaheuristics* (Springer, New York, 2010)
32. G. Gigerenzer, *Gut Feelings: The Intelligence of the Unconscious* (Penguin, New York, 2007)
33. G. Gigerenzer, C. Engel, *Heuristics and the Law* (MIT, Cambridge, 2006)
34. T. Gilovich, D. Griffin, D. Kahneman, *Heuristics and Biases: The Psychology of Intuitive Judgment* (Cambridge University Press, Cambridge, 2002)
35. J. Hirsh, J. Peterson, Extraversion, neuroticism, and the prisoners dilemma. Pers. Indiv. Differ. **46**(2), 254–256 (2009)
36. M. Hsu, M. Bhatt, R. Adolphs, D. Tranel, C. Camerer, Neural systems responding to degrees of uncertainty in human decision-making. Science **310**, 1680–1683 (2005)
37. E. Hunt, The role of intelligence in modern society. Am. Sci. July/August, 356–368 (1995)
38. T. Judge, C. Higgins, C. Thoresen, M. Barrick, The big five personality traits, general mental ability, and career success across the life span. Person. Psychol. **52**, 621–652 (1999)
39. M. Kane, D. Hambrick, A. Conway, Working memory capacity and fluid intelligence are strongly related constructs: Comment on Ackerman, Beier, and Boyle. Psychol. Bull. **131**, 66–71 (2005)
40. R. Kaplan, D. Norton, *The Balanced Scorecard: Translating Strategy into Action* (Harvard Business Press, Boston, 1996)
41. P. Kyllonen, R. Christal, Reasoning ability is (little more than) working-memory capacity? Intelligence **3**, 1–64 (1990)
42. S. Lewandowsky, K. Oberauer, L.-X. Yang, U. Ecker, A working memory test battery for MatLab. Behav. Res. Meth. **42**(2), 571–585 (2011)
43. A. Lo, D. Repin, The psychophysiology of real-time financial risk processing. J. Cognit. Neurosci. **14**(3), 323–339 (2002)
44. A. Lo, D. Repin, B. Steenbarger, Fear and greed in financial markets: A clinical study of day-traders. Am. Econ. Rev. **95**(2), 352–359 (2005)
45. B. Lucey, M. Dowling, The role of feelings in investor decision-making. J. Econ. Surv. **19**(2), 211–237 (2005)
46. K. Morsanyi, S. Handley, How smart do you need to be to get it wrong? The role of cognitive capacity in the development of heuristic-based judgment. J. Exp. Child Psychol. **99**, 18–36 (2008)
47. K. Oberauer, H.-M. Süß, O. Wilhelm, W. Wittman, The multiple faces of working memory: Storage, processing, supervision, and coordination. Intelligence **31**, 167–193 (2003)
48. T. Offerman, J. Sonnemans, What's causing overreaction? An experimental investigation of recency and the hot-hand effect. Scand. J. Econ. **106**(3), 533–554 (2004)
49. B. Roberts, N. Kuncel, R. Shiner, A. Caspi, L. Goldberg, The power of personality: The comparative validity of personality traits, socio-economic status, and cognitive ability for predicting important life outcomes. Perspect. Psychol. Sci. **2**, 313–345 (2007)
50. J. Rust, J. Miller, R. Palmer, Behavior of trading automata in a computerized double auction market, in *Double Auction Markets: Theory, Institutions, and Laboratory Evidence*, ed. by D. Friedman, J. Rust (Addison Wesley, CA, 1993), pp. 155–198
51. J. Rust, J. Miller, R. Palmer, Characterizing effective trading strategies: Insights from a computerized double auction tournament. J. Econ. Dyn. Contr. **18**, 61–96 (1994)
52. H. Shefrin, *Beyond Greed and Fear: Understanding Behavioral Finance and the Psychology of Investing* (Oxford University Press, New York, 2007)
53. V. Smith, *Papers in Experimental Economics* (Cambridge University Press, Cambridge, 1991)
54. E. Weber, E. Johnson, Decisions under uncertainty: Psychological, economic, and neuroeconomic explanations of risk preference, in *Neuroeconomics: Decision Making and the Brain*, ed. by P. Glimcher, E. Fehr, A. Rangel, C. Camerer, R. Poldrack (Academic, New York, 2008), pp. 127–144

55. M. Wellman, A. Greenwald, P. Stone, *Autonomous Bidding Agents: Strategies and Lessons from the Trading Agent Competition* (MIT, Cambridge, 2007)
56. M. Xia, J. Stallaert, A.B. Whinston, Solving the combinatorial double auction problem. Eur. J. Oper. Res. **164**, 239–251 (2005)

Chapter 3
Application of Intelligent Systems for News Analytics

Caslav Bozic, Stephan Chalup, and Detlef Seese

Abstract The chapter starts with an overview of existing text mining systems whose main purpose is predicting equity price movements on the financial markets. In general, these systems transform the input text to a so-called sentiment score, a numerical value equivalent to the opinion of an analyst on the influence of the news text to the further development of the regarded stock. In the second part it is explored how the sentiment score relates to some of the relevant macroeconomic variables. It is suggested that raw sentiment score can be transformed to reveal sentiment reversals, and such transformed indicator relates better to future returns. As an example the project FINDS is presented as an integrated system that consists of a module that performs sentiment extraction from the financial news, a benchmark module for comparison between different classification engines, and a visualization module used for the representation of the sentiment data to the end users, thus supporting the traders in analysing news and making buy and sell decisions.

3.1 Introduction

News articles make a very important information source for traders. News stories reach a huge number of people, and they can initiate massive market movements, like panic selling or massive buying, but they can also lead to more subtle market movements. Until recently it was mainly the task of human analysts to determine how positive or negative a news story is for a subject company. In general, we call such a positivity or negativity measure “text sentiment”.

C. Bozic • D. Seese (✉)
Institute AIFB, Karlsruhe Institute of Technology, Karlsruhe, Germany
e-mail: bozic@kit.edu; detlef.seese@kit.edu

S. Chalup
School of Electrical Engineering and Computer Science, The University of Newcastle, Australia
e-mail: stephan.chalup@newcastle.edu.au

M. Doumpos et al. (eds.), *Financial Decision Making Using Computational Intelligence*, Springer Optimization and Its Applications 70, DOI 10.1007/978-1-4614-3773-4_3,
© Springer Science+Business Media New York 2012

With a rise of algorithmic trading volume in recent years, the need for quantifying qualitative information in textual news and incorporating that additional information into new trading algorithms emerged. This task has to be done on a vast amount of data and in millisecond frequency range, so these requirements render human analysts less useful and machines have to take over the task of quantifying text sentiment.

There are two paths of using quantified text sentiment information—one is feeding it to the machinery that executes trades automatically, and there it is combined with other indicators and used by an algorithm to execute trades most efficiently. The alternative approach is to present this sentiment information to traders visually in a comprehensible way, so the news articles are summarized and the important ones can be filtered. This second approach offers support to professional traders in making trading decisions and relieve them from the burden of following all news releases, including those unimportant and uninteresting.

In this chapter we will give an overview of the most important systems with this goal found in the literature and then describe the system developed within the Karlsruhe Institute of Technology (KIT) and its unique way of using Neural Networks and Support Vector Machines methodology to analyse news articles and support traders in their decisions.

3.2 Present Results

In the past decade different systems and methods appeared in the literature that try to solve the task of automatic news analysis. They use text mining of publicly accessible financial texts in order to predict market movements. They employ different machine learning approaches, define important features in text in a different way, and use different and often incomparable criteria for performance measurement. Although some of them do not explicitly use the term "text sentiment", if we observe text mining methodologies as transformations that assign a numerical value to every textual string, we can refer to that numerical value as a sentiment score. All these publications have at least implicit statements about the predictive power of the specific sentiment score on, for example returns or volatility.

In the following sections we will describe these systems, always following the similar structure. We start with a brief overview and a list of publications where the system was described. Then a source of news data is described, followed by a description of a quantitative data from markets that was used and a definition of target values or classes. Afterwards, we present approaches to document representation, followed by employed classification methodologies. Finally, methods of performance assessment are described, and a section closes with a short discussion, where appropriate. If not stated otherwise, we describe the most recent version of the system, supposing it to be most advanced and best performing.

3.2.1 System LOLITA for Finance

Forerunner of all systems of this type that can be found in the academic literature was a financial text summarization system based on natural language engineering system LOLITA (Large-scale, Object-based, Linguistic Interactor, Translator, and Analyser). It was developed at the University of Durham, and it is first mentioned in [42]. In the same year the first proposal to use this system for financial text analysis came as a publication [14]. The financial application of the system was described in a series of publications [12, 13, 15, 16]. The LOLITA system translates text into a semantic network representation, and the built-in templates for financial activities extract specific events that are expected to have an influence on the market movements. Identified and implemented built-in templates can recognize company takeover, merger, privatization, new issue of shares, new stake, dividend announcement, overseas listing, and bankruptcy. Further analysis and relating extracted event properties to a quantitative data from stock markets were out of scope of this work. That is why it is noted as a forerunner only.

3.2.2 System PDR

The first prototype of the system that uses data mining techniques to predict market movements was presented in the master thesis [34]. The extended version was described shortly afterwards in [66]. The system originates from Hong Kong University of Science and Technology, and it employs methodology of probabilistic datalog rules (PDR) to predict daily change of Hong Kong's Hang Seng Index (HSI). The improved system is presented in the subsequent publications [9, 10], and finally it is in detail described in the PhD thesis [11].

The analysed text is built by merging all articles published during 1 day in all relevant data sources. The list of data sources contains 41 daily changing documents from financial web sites: Wall Street Journal (www.wsj.com), Financial Times (www.ft.com), CNN (www.cnnfn.com), International Herald Tribune (www.iht.com), and Bloomberg (www.bloomberg.com). There is an elaborated procedure for selecting relevant data sources by comparing its performance on the training dataset.

The textual data is aligned with daily values of the Hang Seng Index. The total of 120 training days are split into three classes, "up", "steady", and "down", according to daily change of HSI. Changes between -0.5% and $+0.5\%$ are considered as "steady", while changes of more than 0.5% in positive and negative direction are considered as "up" and "down", respectively. The threshold value used in the first version of this system was 0.3%.

The underlying vocabulary is handcrafted, but unfortunately the complete list can be found only in [34] and contains 125 phrases, for example "hang seng index surged", "shares rose", "political worries", or "dollar edged lower". According to

[11], later vocabulary contains 392 words and phrases, selected from the list of 790 suggested by experts. The first version of the system matched only the complete vocabulary phrases in the analysed text, what is in subsequent version changed to word-level matching, but no version uses word stemming. The document vector is built using term frequency with either inverse document frequency (IDF) or category frequency (CF).

The system uses PDR to build rules that assign scores to text documents. The set of rules is built for each target class, so each analysed document is assigned three distinct scores for “up”, “steady”, and “down”. The scores are compared to certain thresholds to obtain a categorical result (zero or one) for each of the classes. If the result is inconsistent and the text is assigned to none or more than one class, the conflict resolution is conducted. Three different conflict resolution methods are proposed: based on the simple maximum likelihood, on the maximum deviation from the corresponding threshold, and on the nearest neighbour.

As the final step, the system predicts closing value of HSI HSI_t using classification result CP and previous day’s closing value HSI_{t-1}. Three delta values δ_{up}, δ_{steady}, and δ_{down} are calculated from training data for each class separately. The delta is defined as an average difference of HSI values in 2 consecutive days. N_c represents number of days in training set falling into particular class c:

$$\delta_c = \frac{1}{N_c}\sum_{t\in c}(HSI_t - HSI_{t-1}).$$

The predicted closing value HSI'_t is then calculated by adding the appropriate delta for the predicted class CP to the previous HSI closing value:

$$HSI'_t = HSI_{t-1} + \delta_{CP}.$$

Performance evaluation is consistently done using overall accuracy as the main performance measure. The best reported result in [11, p. 161] is 51.9% accuracy for predicting direction of HSI change, and it is achieved with majority voting among multiple data sources and calculated for 79 trading days. Besides HSI, system described in [66] predicts also Dow Jones Industrial Average, Nikkei 225, Financial Times 100 Index, and Singapore Straits Index. The best reported accuracy there is 46.7% for the Financial Times 100 Index. These evaluations are done using 60 trading days.

In one short passage of [10] the same result is expressed in a different manner. The authors analyse the probability that the outcome of classification would have reached certain accuracy if the prediction was totally random. Even for the worst prediction accuracy in the experiment of 40.2%, they could refute the null hypothesis that the same result could come from random guessing with 99.82% confidence.

It is worth mentioning that [66] compares the system based on PDR with two further systems based on k-nearest neighbour learning and neural networks. For the feed-forward neural network with 423 input neurons (one for each of the features), 211 neurons in hidden layer, and three output neurons, trained by back propagation, they report the best accuracy of 43.9% for HSI, calculated on 40 trading days. Unfortunately, further information about the neural network (for example activation function and determination of one resulting class) was not included in the publication.

The same article employs a trading strategy for performance evaluation. The articles published overnight are used as an input to the system, as well as the closing index value from the previous day. This allows to predict the direction of index change before the market opening. According to the trading strategy, stocks representing index or appropriate options are bought if the resulting class is "up", and sold if the resulting class is "down". If the system classifies text to "steady", nothing is done. The reported profit for the Dow Jones Industrial Average is 7.5% for 60 trading days where actual trading according to the strategy takes place on 40 days.

As the authors of [41] well noted, the trading strategy is based on the assumption that at the moment the market opens one can on average sell and buy at yesterday's closing price. They state

> However, this approach is not acceptable, because - taken that HKUST (system developed at Hong Kong University of Science and Technology) recognizes the majority of positive respectively negative developments correctly - it is unlikely that in the case of a positive (negative) prediction one can buy (sell) at yesterday's closing price, but in average certainly at a higher (lower) price.

Maybe even more important is the fact that the authors of this trading strategy do not use real changes in index value for the evaluation, but instead they use the fixed thresholds defined for each of the classes, in this case +0.5 and −0.5%. This means that the conclusion about profitability of the trading strategy is to be taken with reservations.

3.2.3 System DC-1 for Finance

Originally created for a fraud detection, in [22] the authors present an application of the DC-1 system for the task of news monitoring. The paper in which it has been presented as a main contribution has a high-level definition of the general monitoring task which is a part of data mining. The authors propose a framework for evaluating performance of the monitoring systems based on a modified receiver operating characteristic (ROC) curve (for more on ROC, see [21, 46]). They use news monitoring task as an illustration of their framework's application.

The evaluation dataset contains news messages published in the 3-month period. The exact period, the total number of messages, and the source of the messages were not disclosed. There are about 6,000 different companies detected in the dataset. The news stream was aligned with (most probably daily) stock prices, and the event of interest was a surge or a drop in the stock price greater than 10%. According to the proposed framework, the scoring function has to be defined. The scoring function represents the utility of detecting the event in a certain point in time, and it is set to 1 if the event of a price change was recognized during the previous day, or on the day of the price change before 10:30 a.m. Otherwise, the scoring function is set to 0.

Each story was lexically analysed and reduced to its constituent words. The words are stemmed and those from a stop word list are removed from the further analysis. The ordered pairs of two adjacent words (bigrams) were also used in the final vector representation of the news story. The stories preceding substantial change in price (either positive or negative) are labelled as positive examples. This means that all subsequent performance analyses do not distinct between positive and negative change in stock prices, so any direct economic interpretation of the results is difficult.

For performance evaluation, the authors use a modified version of ROC named activity monitoring operating characteristic (AMOC). The variables represented on the axis of this graphical representation are modified to be in accordance with the sequential nature of the monitored data streams. So the x axis represents false alarm rate, normalized to some time interval, and y axis shows average score, according previously defined scoring function. The results presented in the paper are calculated using tenfold cross-validation. The AMOC curve of the system is presented in the publication, and some of the specific points read from the graph are, for example (0.1, 0.76), (0.2, 0.89), and (0.5,0.97). The points of the AMOC curve are determined by varying the threshold of the event detection trigger.

The authors introduce two orthogonal distinctions for classifying approaches in stream mining: the first distinction separates "profiling" on the one end, where only one class of the events is modelled, and all the events that substantially differ from this profile are considered events of interest and "discriminating" on the other end, where both classes are modelled with specific attention to differences between classes. The other distinction is made between a "uniform" and an "individual" approach. In the "uniform" approach only one model for all streams is built, while "individual" approach models each data stream individually. The authors show that the individual approach does not bring much improvement in the news monitoring case study, although introducing more complexity.

3.2.4 System Ænalyst

Started at the University of Massachusetts Amherst, the system under the name of Ænalyst aims at identifying news stories that strongly correlate with significant shifts in the stock price. The project proposal is published in [31], while the actually implemented system was described in [32, 33]. The system uses news messages collected from Biz Yahoo! and modified time series of stock prices transformed to trends by piecewise linear regression, and tries to mine these two types of data concurrently.

The source of the data consists of news stories published via Biz Yahoo! in the period from 15 October 1999 to 10 February 2000. The authors report 38,469 news stories mentioning 127 companies, and the stories are already tagged with the company names they mention. The selection of the companies is done according to the specific criteria: 15 stocks with the largest increase on any day, 15 with the largest decrease on any day, and the certain number of stocks from the three lists published periodically that mention companies that are most actively traded, gainers, and losers. This kind of ex post selection of the companies comprising the dataset makes the trading strategy evaluation results biased in the way that is not attainable in the real market conditions (see [41]).

The quantitative part of the dataset contains stock prices for these 127 companies during the opening hours of the exchanges, with the 10 min resolution. The period is equivalent to the period of the news dataset. "Piecewise linear regression" is used to represent the time series as an array of trend lines. These trends are clustered according to the slope and confidence (squared error) of each linear segment, and after the inspection of the distribution, it is decided that a simple binning procedure can be used. The trends are divided into five classes: "surge" ("plunge"), if the slope is greater (less) than 75% of the maximum (minimum) slope, "slight+" ("slight−") if the slope is between 50 and 75% of the maximum (minimum) observed slope, and "no recommendation" otherwise.

The authors propose named entities, linked objects, and subsumption hierarchies as approaches for text representation (see [31] for more details). Though, the evaluation in subsequent publications [32, 33] is done for bag of words approach only. Bag of words is an approach to document representation where unordered collections of words are used, assuming conditional independence and disregarding the order of words. The system builds a language model for all messages that precede the beginning of the particular trend for a certain period. If t is the moment in which the particular trend starts, then all the messages published between $t-h$ and t are considered to have influence on the appearance of the observed trend. These messages are used to build the language model for the class of the observed trend. The variable h represents the chosen time horizon, and it is varied from 5 to 10 h during the evaluation. Authors use additional approach of aligning only the stories published contemporaneous with the observed trend, with the note that this kind of model is rather explanatory than predictive.

After all of the news stories are associated with trends, and trends are divided into classes, classification is done by building language models with an assumption of words independence, effectively arriving at a naïve Bayesian classification. Since the news analytics and news classification is an on-line task, where used language can change, we need to assure that these new terms have non-zero probability; otherwise, it would drive the model's probability to zero value. This is done by a linear back-off algorithm, where the conditional probability of word ω under the model M_t is given by formula

$$P(\omega|M_t) = \lambda_t P_{ml}(\omega|M_t) + (1-\lambda_t)P(\omega|GE),$$

where P_{ml} represents the original probability under the certain language model that possibly can be equal to zero and GE denotes background of general English language model. The parameter λ_t is calculated according to formula

$$\lambda_t = \frac{N_t}{N_t + U_t}$$

with N_t denoting total number of tokens and U_t denoting the number of unique tokens in the model M_t.

The evaluation of the prototype is done using two approaches: a detection error tradeoff (DET) curve, similar to ROC, and a trading strategy. DET is a graphical representation of the classification performance that plots miss rate against false alarm probability. Tenfold cross-validation was used to calculate the results. The plot contains points like (0.5, 90) and (15, 60) being percentages of false alarms and miss rates, respectively. This means that the system can achieve 10% recall with false alarm rate of 0.5% in the task of predicting exact class of the trend up to 10 h ahead.

The previous evaluation works with pairs (t, D), where t represents a trend and D a document, so one document can be associated with more than one trend (in fact it is associated with all trends that follow the particular document in next 10 h). For a practical evaluation, one need to introduce conflict resolution and determine only one prevailing resulting class for the each document. This is done in the evaluation that uses trading strategy. Different language models are built for each stock using the data from the first 3 months of the dataset, while the rest of the dataset was used for evaluation. According to the most probable class, the trading strategy should buy or short-sell the stock worth of 10,000 USD. The budget is considered unlimited, and the transaction costs are considered to be zero. The reported total gain was 21,000 USD within the whole dataset, with, for example in the case of Yahoo! stocks 0.5% gain per transaction. With changing the time window for associating documents and trends in the range from 1 to 10 h, the average profits varied from 290 USD to 2,360 USD.

3.2.5 System from University of California, San Diego (UCSD)

News analytics prototype developed at the University of California in San Diego and financed by US Air force was described in a project report [26] and somewhat more detailed later in [25]. In the first version, it uses the same dataset as prototype Ænalyst and an original approach to aligning text and price values. The naïve Bayesian classifier is used for text classification.

The dataset contains 25,087 articles about 30 companies of the Dow Jones Industrial Average (DJIA) index in the period from 26 July 2001 to 16 March 2002. After excluding the articles published outside of business hours and excluding the articles that correspond to the periods where price data is incomplete or missing, the authors report 12,437 news articles. The influence of the published article to the market is modelled by the "window of influence", thus creating the model similar to event study. Window of influence is the period in which the published article should have the influence on the stock price of the company that is mentioned. Different lengths of the window of influence up to 30 min are evaluated, and this is done for the two cases: when the window of influence ends at the moment of news article publication and when the window of influence starts at the moment of news article publication.

The price data is collected on the 1-min level. The classes are determined according to the β and index adjusted change m in the stock price within the window of influence. The price data is transformed according the following formula:

$$m(t_1,t_2) = \frac{\delta p_s(t_1,t_2)}{\beta} - \delta p_i(t_1,t_2),$$

where t_1 and t_2 represent endpoints of the window of influence, $\delta p_s(t_1,t_2)$ is the change in the observed stock price and $\delta p_i(t_1,t_2)$ is the change of the index value during the window of influence. The parameter β determines relation between the price of the particular stock and the market as a whole. The value of DJIA index is chosen as a proxy for market performance in this case.

According to this value, the target classes are labelled as "up", "normal", and "down". If the value of m is less than the negative threshold ρ_{negative}, the target class is "down", if the value of m is greater than positive threshold ρ_{positive} the target class is "up", and otherwise the target class is "normal". The values for ρ_{negative} and ρ_{positive} are determined in such a way that the frequencies of the classes "up", "normal", and "down" are approximately 25%, 50%, and 25%, respectively. The classification is performed using a naïve Bayesian classifier.

Depending on the prior distribution of classes the trivial classifier that always predict the class with highest prior can have better overall accuracy, although not being economically useful, as noted by the authors. That is why they suggest two performance measures based on the economic utility of the classification. One is based on the β and index adjusted price change m, which authors call "normalized economic value estimate" (NEVE). They also mention the major drawbacks of this measure by saying

…the meaning of NEVE is not easy to understand, nor even its unit is clear.

The second performance measure the authors suggest is the average profit per trade achieved in the specific trading strategy. According to the strategy, at the beginning of the window of influence one buys (sells) 1,000 USD worth of shares of the mentioned company and sells (buys) $\beta\times$ 1,000 USD worth of the index if the predicted class was "up" ("down"). At the end of the window of influence, this position is cleared.

The best performance according to both criteria is achieved using the window of influence with the length of 20 min before the publication of the news article. The reported profit per trade is 1.0063 USD and NEVE value of about 0.17. The question of economic applicability of the trading strategy in real conditions stays open since it seems that it incorporates some actions that have to take place before the classification of the news article is performed.

3.2.6 System from the Chinese University of Hong Kong (CUHK)

The prototype developed at the Chinese University of Hong Kong is the first system which analyses news stories published by Reuters, thus achieving to collect about 600,000 news items for the training and evaluation. It is mostly composed of already existing natural language processing (NLP) and machine learning (ML) components, combined in a new manner to allow for parallel mining of text and multiple time series. The system is described in two publications [23, 24].

These 600,000 news articles were published between 1 October 2002 and 30 April 2003, and the first 6 months were used for training, while the last month was used for the out of sample evaluation. The news articles are tagged with company names mentioned, so the system uses all messages in that period that mention 33 components of Hang Seng Index.

The time series of daily prices is transformed using piecewise linear regression, similar to the one in [33]. It is just not clear whether both opening and closing prices represent points in time series, or just one type of daily price, and in the latter case, which one. All trends generated this way are characterized by slope m and goodness of fit R^2 and clustered into three classes: “rise”, “steady”, and “drop”. The cluster with greatest average slope is selected for the class “rise”, the cluster with the lowest average slope is selected for class “drop”, while the third cluster becomes class “steady”.

The documents are represented by a bag of words approach where all the words are stemmed, and with punctuation, stop words, web links, and numbers excluded. The features produced that way are weighted according to the tf$*$idf schema.

The authors chose to align the documents with contemporaneous trend only, not allowing for delayed influence of the news to stock prices. They back up this decision by effective market hypothesis, although in later evaluation of the system’s performance, they use trading strategy and show its profitability, thus violating the previous principle. The improvement and the novel approach is to once more cluster the documents falling into the categories “rise” and “drop”, thus creating two subclusters each, and then removing one subcluster from the “rise” category which is more similar to the documents of the “drop” category, and vice versa—removing one of two subclusters from the “drop” category that is more similar to the documents of the “rise” category. This provides two more distant classes of documents for “rise” and “drop” categories, creating less confusion during the learning phase.

The model learning is done using SVMlight support vector machine software [63]. Two SVMs are trained, for “rise” and “drop” class. If the news article is classified positive by both of the classifiers, it is labelled as ambiguous, effectively falling into the “steady” class.

The evaluation is performed by applying a trading strategy to the prices and news data from April 2003. According to the strategy, the shares are bought (sold short) in the moment of publication of the news article that is classified as “rise” (“drop”), and the position is cleared after 24 h. If the news article is published during the holidays, the shares are bought (sold short) immediately after the stock market opens on the first business day. Since the authors have only daily data at disposal, it is not clear which daily price they use for the market simulation. The best performing version of the system achieves a rate of return of 0.1985, but the total number of trades was not disclosed. The usual assumption of zero transaction cost applies also.

3.2.7 System AZFinText

One of the recent systems that is still in development bears the name of AZFinText. It is s built upon one version of Arizona Text Extractor (AzTeK) which is described in [64]. The AZFinText system aims at predicting price of particular stock after a 20-min period by analysing news published via Yahoo! Finance. The system is presented consistently and with slight constant improvements in a long row of publications [49–57].

As the only data source the system uses articles from Yahoo! Finance, tagged with the company they relate to. Texts published outside of stock markets opening hours, as well as those published less than 1 h after markets opening and less than 20 min before markets closing, are filtered out. Excluded are also messages that are less than 20 min apart. The final dataset used for analysis contains 2,802 articles from 5 trading weeks that relate to 484 of the S&P 500 companies. The goal is to predict the exact equity price exactly 20 min after the news text has been published.

The input text is preprocessed using four different approaches: bag of words, noun phrases, proper nouns, and named entities. Only phrases with three or more instances per document are represented with one in resulting binary vector, while the other phrases are represented as zero. The binary document vector created in this way, together with equity price in the moment of article publication, is used as an input to machine learning engine.

Machine learning part of the this prototype is built on SVM trained using sequential minimal optimization as described in [45] and using linear kernel. As an output from the training step, SVM estimates parameters of a regression by assigning weights to each of the input terms. The dependent variable of this regression represents the predicted equity price.

In [57] the authors use additional inputs to the machine learning engine. It represents sentiment assessment of an analysed article, produced by OpinionFinder. OpinionFinder analyses text and classifies it as rather “subjective” or rather "objective", and its tone as "positive" or "negative". In both classifications, the classifier can also abstain from classification. The article reports that just adding subjectivity and tone information leads to deteriorated result, but on the other hand, results improve when the prediction is done only on subset of articles classified by OpinionFinder as "subjective" or "negative".

For performance evaluation there are three approaches consistently used through publications. All of them employ tenfold cross-validation. The first measure of accuracy is equivalent to the mean squared error between real and predicted equity price. The lowest reported value is 0.03407 on 2,620 data points when named entities preprocessing approach is used. Unfortunately the nature of this measure and its dependence on absolute value and range of target variables makes it unfit for further comparisons.

The second approach considers observing the direction of change only. If a predicted price and a realized price are both higher or both lower than the price at the moment of prediction, the prediction is considered correct. This is equivalent to the overall accuracy for the case of two-class classification. The best reported value in this category is 58.2% which is achieved using a proper noun preprocessing approach on the whole test set of 2,809 data points, with higher accuracy of 59% if calculated for 61 subjective text only and even higher for the case of training the SVMs separately for different industry sectors, reaching 71.18%.

The third approach is using a trading strategy that follows ideas given in [39]. It invests by buying stocks which have predicted an increase of more than 1% and short-selling a stocks which have predicted a decrease of more than 1%. The position is in both cases cleared after 20 min. The best reported profit 8.5% is reported when using SVMs trained separately on different industry sectors in [54]. It is, though, not clear how many trades this scenario contains. The number of trades is only reported in [49], where the best profit of 3.6% is achieved in 108 trades with named entities preprocessing model used.

3.2.8 Other Systems

Further prototypes include the system called news categorization and trading system (NewsCATS) described in [40, 41]. NewsCATS is a high-frequency forecast system that classifies press releases of publicly traded companies in the USA using a dictionary that combines automatically selected features and a handcrafted thesaurus. For classification the authors use SVMs with polynomial kernels.

One of a few systems that employ neural networks is described in [35, 36]. The system proposed in the first publication uses only the volume of posted internet stock news to train a neural network and predict changes in stock prices, so we cannot consider the system proposed there as a real text mining system. As an extension, the second work [36] employs NLP techniques and a handcrafted dictionary to predict stock returns. They use a feed-forward neural network with five neurons in the input layer, 27 in the hidden layer, and one output neuron. Since only 500 news items were used for the analysis, no statistical significance of the results could be found.

Another group of publications contains works that do not primary attempt to prove economical relevance of published text by evaluating specifically tailored trading strategies, but rather to find statistically relevant relations between financial indicators and sentiment extracted from the text.

The authors of [1] use naïve Bayes and SVM classifiers to classify messages posted to Yahoo! Finance and Raging Bull and determine their sentiment. They do not find statistically significant correlation with stock prices, but they find sentiment and volume of messages significantly correlated to trade volumes and volatility. In their methodological paper [18], authors offer a variety of classifiers, as well as composed sentiment measure as a result of voting among classifiers. In the illustrative example, they analyse Yahoo! stock boards and stock prices of eight technology companies, but they do not find clear evidence that the sentiment index can be predictive for stock prices.

There are two pivotal articles published in the Journal of Finance. Tetlock in [60] observes Wall Street Journal's column "Abreast of the Market", uses content analysis software General Inquirer together with principal component analysis approach and finds that high pessimism in published media predicts downward pressure on market prices. The authors of [61] succeeded to find that rate of negative words in news stories about certain company predicts low earnings of the company.

While most of the previously described publications focus on predicting price trends of single stock or index, there are publications that aim at determining influence of news releases to volatility. System [62] improves the risk-return profile by exiting the market in case of news that are predicting high volatility, while [48] attempt to classify press releases of German public companies according their influence on volatility of stock prices. There are also systems that aim at predicting foreign exchange rates, for example [44, 65], but we will stay focused on the sentiment extraction systems for equity prices prediction.

From the previous sections and Tables 3.1–3.6 we can notice that existing systems implement various classification methodologies—at the beginning mostly naïve Bayes and decision rules, later more SVMs, and recently just a few that use neural networks. The feature definition procedure and document representation are quite different between systems, what makes comparison of different classification approaches a complicated task.

It is of special interest to compare the performance evaluation of all these systems. Some of performance measures are purely machine learning and statistically oriented, making them less useful for financial setting. The others are based on historical data and trading strategies, but in most of these cases, lack enough evaluation data points or use assumptions that are unattainable in real life. All presented systems have between 40 and about 13,000 evaluation data points, what might be not enough to guarantee stability of the results. Additionally, both systems having more than 10,000 evaluation points include some assumptions that are observable only ex post.

Table 3.1 Properties of the prototype PDR

Prototype PDR		
Features definition	Handcrafted	
Inputs	Text	41 data sources a day
Classifier	Rule-based PDR	
	(k-nearest neighbour)	
	(feed-forward NN)	
Output	Categorical	
Prediction	Index value change	
Frequency	Daily	
Performance	Overall accuracy	51.9%/3 classes
	Trading strategy	7.5%/40 trades
Evaluation data points		40

Table 3.2 Properties of the prototype DC-1 for finance

Prototype DC-1 for finance		
Features definition	Automatic	
Inputs	Text	3 months and 6,000 companies
Output	Categorical	
Prediction	Stock price jump	
Frequency	(Daily)	
Performance	ROC-AMOC	
Evaluation data points		Undisclosed

Table 3.3 Properties of the prototype Ænalyst

Prototype Ænalyst		
Features definition	Automatic	
Inputs	Text	Biz Yahoo!
Classifier	Naïve Bayesian	
Output	Probabilities/categorical	
Prediction	Trend class	
Frequency	10 min	
Performance	DET	
	Trading strategy	0.5%/trade
Evaluation data points		12,174 trades

Table 3.4 Properties of the prototype UCSD

Prototype UCSD		
Features definition	Automatic	
Inputs	Text	
Classifier	Naïve Bayesian	
Output	Categorical	
Prediction	Normalized price change	
Frequency	1 min	
Performance	NEVE	0.17
	Trading strategy	1.0063 USD/trade
Evaluation data points		13,372 articles

Table 3.5 Properties of the prototype CUHK

Prototype CUHK		
Features definition	Automatic	
Inputs	Text	
Classifier	SVM	SVMlight
Output	Categorical	
Prediction	Trend	
Frequency	Daily	
Performance	Trading strategy	0.1985 rate of return
Evaluation data points	1 month daily data for 33 companies	

Table 3.6 Properties of the prototype AZFinText

Prototype AZFinText		
Features definition	Automatic	
Inputs	Text	Yahoo Finance
Classifier	SVM	
Output	Categorical	
	Real	
Prediction	Stock price	
Frequency	1 min	
Performance	Mean squared error	0.03407
	Overall accuracy	71.18%
	Trading strategy	8.5%
Evaluation data points	Tenfold cross-validation on 2,802 articles	

We suggest to bridge this gap by creating a benchmark based on sufficient data to ensure statistical relevance of the results (over 100,000 articles and 600,000 daily returns). We also offer the spectrum of different classification approaches, while keeping the text preprocessing and document representation constant, enabling the effective comparison of classification engines.

Besides the prototypes described in the academic literature, there are systems developed in the industry. They employ methodology that is proprietary and rarely described in the academic literature. Nevertheless, they represent de facto industrial standard, being used by a great number of professional traders. Such services are provided under the name of News Analytics in the product portfolios of companies like RavenPack (see Sect. 3.3.2) and Thomson Reuters (see Sect. 3.3.1).

3.3 Text Sentiment Properties

The publications presented in the previous section showed a relation between extracted sentiment from financial news and equity prices, trade volumes, and volatilities. We will take here the macroeconomic view and explore the influence of such sentiment score to GDP and other macroeconomic variables. We also propose a way of transforming raw sentiment score to expose sentiment reversals and show that the data transformed in this way relate even better to future returns.

3.3.1 Macroeconomic Data

The data used in this research originates from two main sources: the source for quantified news data and the basis for sentiment score comes from Reuters NewsScope Sentiment Engine for historical data, while the public World Economic Outlook (WEO) database [29] of the International Monetary Fund provided the macroeconomic data.

Data constituting the output of the Reuters NewsScope Sentiment Engine (now part of the Thomson Reuters News Analytics product) represents the authors sentiment measure for every English language news item published via NewsScope in the period from 2003 to 2008 inclusive. The measure classifies a news item into one of three categories: positive, negative, or neutral. The probability of the news item falling into each of the categories is also given. Each record represents a unique mention of the specific company, with a possibility of one news item relating to more than one company. In our dataset there are 6,127,190 records. Each news item can have multiple tags called topic codes, and topic codes are grouped into categories. One of the categories represents countries, and it can be used for determining what country is mentioned in the particular news item. Using this tagging feature, average sentiment per country can be calculated. The data is aggregated using the system described in [6] to the level of country and year. The data provided by Reuters are outputs of the classifiers in the form of three probabilities: probability that the analysed news item is positive in sentiment P_{pi}, that it is neutral P_{oi}, or that it is negative in sentiment P_{ni}. We define sentiment score S_i by subtracting probabilities for positive and negative class, and by aggregation we get a yearly sentiment score for a country $S_{y,c}$:

$$S_i = P_{\mathrm{pi}} - P_{\mathrm{ni}}.$$

This way we get panel data with 6 consecutive years and 233 countries and territories. We combine this data with the data for the same period from the WEO database. After excluding countries that are not a part of the WEO database, we get the data about 181 countries. The selected subset of variables from WEO dataset is given in Table 3.7. The estimation of panel regression coefficients on the data is performed with the help of OxMetrics and DPD software [20].

Table 3.8 shows estimated values of coefficients for the OLS panel regression when only contemporaneous sentiment is used as independent variable. The star notation of the statistical significance of the results is explained in Table 3.12. We find positive and significant relation of the average sentiment of all news items published in 1 year and mentioning a country, with that country's GDP in that

Table 3.7 Selected variables from WEO dataset

Variable description	Units	Scale
Gross domestic product, constant prices (change)	Percent change	
Gross domestic product, current prices	US dollars	Billions
Gross domestic product per capita, current prices	US dollars	Units
Gross domestic product based on purchasing-power-parity (PPP) valuation of country GDP	Current international dollar	Billions
Gross domestic product based on purchasing-power-parity (PPP) per capita GDP	Current international dollar	Units
Gross domestic product based on purchasing-power-parity (PPP) share of world total	Percent	
Investment	Percent of GDP	
Gross national savings	Percent of GDP	
Inflation, average consumer prices	Index	
Inflation, average consumer prices (change)	Percent change	
Inflation, end of period consumer prices	Index	
Inflation, end of period consumer prices (change)	Percent change	
Import volume of goods and services (change)	Percent change	
Export volume of goods and services (change)	Percent change	
Unemployment rate	Percent of total labour force	
Population	Persons	Millions

Table 3.8 Estimated coefficients, contemporaneous sentiment score as an independent variable

Variable description	Coefficient
Gross domestic product, constant prices (change)	1.5138**
Gross domestic product, current prices	44.3611**
Gross national savings	6.2975
Import volume of goods and services (change)	4.6756
Export volume of goods and services (change)	1.8859
Population	1.1836***

year, and with that country's GDP change compared to the previous year. Thus, the country with more positive news published in a year will have greater GDP and more positive GDP change by the end of the year.

This prediction goes beyond the time span of 1 year, as the data from Table 3.9 suggests. It shows that the sentiment score in 1 year can be significantly and positively related to GDP, GDP change, gross national savings, and import volume for the following years, up to 3 years ahead. The negative estimations for the coefficients in Table 3.9, like relation to export volume, or import volume 3 years ahead, are all not significant, so we cannot make any statements on that relations.

Table 3.9 Estimated coefficients, lagged sentiment score as an independent variable

Variable description	S	$S(t-1)$	$S(t-2)$	$S(t-3)$
Gross domestic product, constant prices (change)	2.8990**	1.3739	1.6556*	0.9656
Gross domestic product, current prices	32.6456	67.7682*	52.5004	93.2768***
Gross national savings	14.6525*	14.6006**	8.5230*	6.2042**
Import volume of goods and services (change)	13.8133**	12.5976**	0.4568	−2.9750
Export volume of goods and services (change)	1.4063	−4.2891	2.9645	−0.9618
Population	0.9589***	0.4708	1.0382**	1.2311***

3.3.2 Sentiment Reversals

Since the scientific exploration of text mining and sentiment extraction in finance lasts for over a decade, as noted in [7]

> … simple "buy on the good news" and "sell on the bad news" strategies won't likely generate significant alpha as news analytics become more widely adopted.

That is one of the reasons for the great number of publications in recent years which try to find novel approaches and exploit sentiment scores provided by existing systems.

One of the recent publications, with an approach similar to ours, is [30]. The author uses sentiment reversals as buy signals and proves the performance of his approach by simulating a set of portfolios over the time frame 2000–2008. The source of sentiment data is Dow Jones News Analytics historical database, where the sentiment scores have been assigned to news articles using RavenPack proprietary software. Each story is assigned a positive or a negative sentiment indicator as a value of $+1$ and -1, respectively. Each story is also related to the company mentioned in the text. This approach builds one time series of sentiment indicators for each company. The author uses an average value of the sentiment time series over a certain time window to define sentiment reversal. Sentiment reversal is, in fact, defined as an end of the period in which the running window average sentiment has a constantly positive (negative) value. Graphically, this corresponds to all points where graphical representation of the running window average sentiment intersects the x axis. There are some additional constraints introduced, like minimal duration of reversal period of 30 days, and minimally 30 published stories in this period.

The author's findings suggest that the portfolios built using sentiment reversal-based signals for buying and short-selling the companies' stocks can outperform the market both in bull and bear market conditions.

We explored a similar approach, although using different dataset and basing our performance assessment on more general and theoretical approach described in [6] instead of portfolio simulation.

The question we want to answer here is whether sentiment data transformed by extracting sentiment reversals can be more useful for predicting future returns for a company than the raw sentiment score data. We work with sentiment data provided by Thomson Reuters and evaluate it according the framework described in [6].

Historical database of Thomson Reuters News Scope Sentiment Engine is a source of sentiment scores for all English language news published via Reuters News Scope system since year 2003. For each company mentioned in the news, this database contains the probabilities that the author write about the particular company in rather positive or rather negative context. The sentiment score $S(n,c)$ for news article n and company c is built as a difference between the probability that the mention is positive (Pr_{pos}) and the probability that the mention is negative (Pr_{neg}). Since we want to work with daily data, the sentiment score is averaged within each trading day, thus producing daily sentiment score for a company $S(t,c)$. With T we denote a set of all messages published during the trading day t:

$$S(n,c) = Pr_{\text{pos}}(N,C) - Pr_{\text{neg}}(N,C)$$
$$S(t,c) = \sum_{n \in T} S(n,c).$$

Besides sentiment scores this dataset offers additional metadata. Most important for us are the publication time stamp and the identifiers of all the companies mentioned in the news. We form a subset of all news available in the archive by choosing only those news items related to companies that are constituents of the Russell 3000 Index. The Russell 3000 Index consists of the largest 3,000 US companies representing approximately 98% of the investable US equity market.

The sentiment reversal measure is defined in the following way—for each company a running average sentiment score $S_{\text{avg}}(t,c)$ is calculated for each day t. In the moment of news article publication, instead of raw sentiment score as defined in the previous paragraph, we assign a new value that measures sentiment reversal or sentiment deviation from the average value S_r. The number of total trading days before the current day t is denoted by N_t:

$$S_{\text{avg}}(t,c) = \frac{\sum_{t_l < t} S(t_l,c)}{N_t}$$
$$S_r(t,c) = S(t,c) - S_{\text{avg}}(t,c).$$

As a source of trading data we used Thomson Reuters Tick History database. We extract opening prices for all trading days in 2003 for each company from the Russell 3000 Index. The opening prices are adjusted for dividends and then transformed into log returns. In this way we get open-to-open (R_{OO}) returns for each trading day in 2003 and each Russell 3000 company. The respective equation is given below, where P_{O} represents opening stock price and t represents the current trading day:

$$R_{\text{OO}} = \ln \frac{P_{\text{O}}(t)}{P_{\text{O}}(t-1)}.$$

Table 3.10 Estimated OLS coefficients for raw and transformed sentiment score, R_{OO} as an independent variable

Time lag	Raw sentiment score	Sentiment reversal
t	104.159***	100.850***
$t-1$	22.052**	34.596***
$t-2$	−9.126**	−5.034
$t-3$	−8.236**	0.173
$t-4$	−3.569	−2.887
$t-5$	−2.863	0.910

We next align daily sentiment data (both raw data and reversal measure) with daily returns and build the linear regression to explore relation between these variables. If the observed sentiment measure actually correlates with the future stock returns and if we represent the current day's return as a regression of previous sentiments (as in (3.1)), then the coefficients in front of the text sentiment measures should be significantly different from zero. We estimate regression parameters for linear regression with open-to-open return R_{OO} as a dependent variable using ordinary least squares method. As independent variables we use contemporaneous daily sentiment score $S(t)$, daily sentiment score from day before $S(t-1)$, 2 days before $S(t-2)$, and 3, 4, and 5 days before $S(t-3)$, $S(t-4)$, and $S(t-5)$. This is done with respect to the subject company c, which is represented as an additional parameter in the equation, besides time t. We order all the companies in our dataset according to their market capitalization (total market value of all shares of the company), and divide them into ten equally sized groups. In this way we get the values for ten additional dummy variables dd_1 to dd_{10} (being 1 if the subject company falls into the respective group, and 0 otherwise). We include them into the regression to account for the variations of returns as a result of company's size

$$\begin{aligned} R_{OO}(t,c) = {} & \alpha_0 S(t,c) + \alpha_1 S(t-1,c) + \alpha_2 S(t-2,c) + \alpha_3 S(t-3,c) \\ & + \alpha_4 S(t-4,c) + \alpha_5 S(t-5,c) + \sum_{i=1}^{9} \beta_i dd_i(c) + \gamma. \end{aligned} \tag{3.1}$$

We repeat the estimation of the regression parameters using sentiment reversal measure $S_r(t,c)$ instead of raw sentiment score $S(t,c)$ following (3.2). The results are presented in Table 3.10 which offers a possibility for a comparison of these two approaches:

$$\begin{aligned} R_{OO}(t,c) = {} & \alpha'_0 S_r(t,c) + \alpha'_1 S_r(t-1,c) + \alpha'_2 S_r(t-2,c) + \alpha'_3 S_r(t-3,c) \\ & + \alpha'_4 S_r(t-4,c) + \alpha'_5 S_r(t-5,c) + \sum_{i=1}^{9} \beta'_i dd_i(c) + \gamma'. \end{aligned} \tag{3.2}$$

From Table 3.10 it is visible that both sentiment measures have positive and significant relation to contemporaneous return as well as to the return 1 day ahead. With increasing the lag between sentiment measure and return, this relation becomes less significant for both sentiment measures. Since contemporaneous sentiment does not introduce any incremental forecasting ability, we are more interested in lagged sentiments. Relation between today's sentiment reversal and tomorrow's return is stronger and noticeably more significant than the relation between today's raw sentiment score and tomorrow's return.

The conclusion from the results presented in this section, as well from the previous work, is that sentiment reversal measure can be a very useful dimension in news analytics and that any system aiming in representing the results of the sentiment extraction should not omit representing this parameter.

3.4 Project FINDS

Financial News and Data Service (short FINDS) is a project going on within Information Management and Market Engineering (IME) Graduate School at the KIT with the goal to conduct innovative research on the analysis of quantitative and qualitative information from financial markets [2]. The topics covered are broad and include detecting novelty of the published financial articles employing semantic technologies [37, 38], as well as exploring financial markets' reaction to the introduction of machine readable reports in XBRL format [67]. Contribution to the evidence that published information influence financial markets, taking the perspective of empirical microeconomics, has been brought by Storkenmaier et al. [59] and Riordan et al. [47]. The impact of the public announcements on prices of CO_2 emission permits is investigated in [28]. The research presented in the previous section [3–5] is a part of this corpus, too.

The FINDS system itself consists of three main components—the first component is a NLP and text classification engine which is used to extract sentiment from text documents and news articles. Multiple engines exist, and they are based on Bayesian classifier, support vector machines, and neural networks. The extracted sentiment score is delivered to the second component which is used for performance assessment. Employing the methodology of empirical finance and aligning sentiment data with quantitative data on an extensive dataset, this benchmark can determine relative quality of different classifying engines and help in choosing the appropriate one for the specific task, as described in [6]. The produced sentiment scores are numerical values that summarize the tone of the great amount of news articles for a number of different companies. To render this numerical summarization useful, we developed FINDS visualization component as a third main component of the FINDS system.

3.4.1 Sentiment Extraction

As a source of financial news we use the archive of all news items published via Reuters NewsScope in year 2003. Besides news text this dataset offers additional metadata. Most important for us are the publication time stamp and the identifiers of all the companies mentioned in the news. We form a subset of all news available in the archive by choosing only the news items related to companies that are constituents of the Russell 3000 Index. As mentioned earlier, the Russell 3000 Index consists of the largest 3,000 US companies representing approximately 98% of the investable US equity market.

As a source of trading data we have an access to Thomson Reuters Tick History database. We extract opening and closing prices for all trading days in 2003 for each company from Russell 3000 Index. The opening and closing prices are adjusted for dividends and then transformed into log returns. In this way we get open-to-close (R_{OC}), open-to-open (R_{OO}), close-to-open (R_{CO}), and close-to-close (R_{CC}) returns for each trading day in 2003 and each Russell 3000 company. The respective equations are given below, where P_{O} and P_{C} represent opening and closing stock price, respectively, and t represents current trading day:

$$R_{\mathrm{OC}} = \ln \frac{P_{\mathrm{O}}(t)}{P_{\mathrm{C}}(t-1)}$$

$$R_{\mathrm{OO}} = \ln \frac{P_{\mathrm{O}}(t)}{P_{\mathrm{O}}(t-1)}$$

$$R_{\mathrm{CO}} = \ln \frac{P_{\mathrm{C}}(t)}{P_{\mathrm{O}}(t)}$$

$$R_{\mathrm{CC}} = \ln \frac{P_{\mathrm{C}}(t)}{P_{\mathrm{C}}(t-1)}$$

For training we singled out news about four companies: Apple Computer Inc., International Business Machines Corp., Microsoft Corp., and Oracle Corp. We kept only the paragraph where the subject company is mentioned and four surrounding paragraphs since this amount of surrounding text was proven in previous tests to yield the best performance. The words with only one or two characters are discarded. All other words are stemmed, and their absolute frequencies in the text are calculated. Each of 11,781 distinct words in the training set represents one dimension of the training vector. Each news item represents one training vector. The target value is determined according to the next days open-to-open return of the subject company. To decrease the overlapping between time range of news publication and time range of returns, all the news items published after the closing time of the market (3:30 p.m., local time) are considered to belong already to the next date.

The first classification method that can be trained and then later used for classifying unseen documents is Bayesian classification. It is based on the Bayes theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

If we substitute probabilities of a document falling into a certain class and look for the class with the highest probability, the equation has the form

$$C = \arg \max_{c \in \text{classes}} \prod_{\omega \in D} \frac{P(\omega|c)P(c)}{P(\omega|B)}.$$

With C we denoted the selected class, ω is individual word in a document that we want to classify D, while B represents background distribution of words in a particular language, English in our case. This approach assumes mutual independence of individual words, what is a common assumption in the text classification area. Although this assumption is not strictly correct for most of the languages, there are evidence that this simplification has little influence on practical results [19].

The second classifier is based on support vector machines. As a kernel function it uses radial basis function in the form

$$RBF(x_i, x_j) = \exp -\gamma ||x_i - x_j||^2, \gamma > 0.$$

To find the best performing combination of parameters γ and C, which is the penalty parameter of the error term, we perform grid search. Different combination of these two parameters are tested using fivefold cross-validation on the training dataset. The resolution is gradually increased, so we generate the accuracy contour and can select the best values for the parameters. For the detailed discussion on support vector machines applications in text classification and applications of kernel methods in finance, the reader is advised to consult publications [8, 63].

The third system uses a feed-forward neural network with an input layer, two hidden layers, and an output layer. The size of the input layer depends on the properties of input text, and it is defined by the number of distinct words in training dataset. In our case the number of neurons in the input layer is 11,781. The two hidden layers consist of 16 and 8 neurons, while output layer has one or two neurons, depending on the version of neural network. We compared two different versions of neural networks. They differ in structure and training procedure:

- Version 1 has one neuron in the output layer, and the output value of that neuron is the value of future return.
- Version 2 has two neurons in the output layer. During the training, the neurons can take only one of two values—zero or one. The first neuron is set to one if the future return is positive; the second neuron is set to one if the future return is negative.

3.4.2 Benchmark

Each of 107,266 news items published via Thomson Reuters in the year 2003 that mentions any of the Russell 3000 companies is classified. The paragraphs with the mention of the subject company and the four surrounding paragraphs are singled out,

Table 3.11 Estimated OLS coefficients with R_{OO} as an independent variable

	Bayes	SVM	NN v1	NN v2	Reuters NSE
$S(t-1)$	33.22*	44.81*	13.69*	8.96	21.87**
$S(t-2)$	−10.63***	−10.52	−6.36***	−4.07**	−9.37**
$S(t-3)$	7.59***	0.840	10.78***	−0.80	−8.79**

the words with only one or two letters are discarded, and the other words stemmed. This word vector is fed to the Neural Network predictor, and as an output we get the text sentiment.

All the text sentiment results for one company and 1 day (in this case the next day starts already with closing the market—3:30 p.m. local time) are averaged and aligned with the corresponding return for the same company and same date:

$$R_{OO}(t,c) = \alpha_0 S(t,c) + \alpha_1 S(t-1,c) + \alpha_2 S(t-2,c) + \alpha_3 S(t-3,c) + \sum_{i=2}^{10} \beta_i dd_i(c) + \gamma. \tag{3.3}$$

At this point we need the way to determine the predicting power of the text sentiment measure. If the observed text sentiment measure actually correlates with the future stock returns and if we represent the current day's return as a regression of previous sentiments (as in (3.3)), then the coefficients in front of the text sentiment measures should be significantly different from zero. We estimate regression parameters for linear regression with open-to-open return R_{OO} as a dependent variable using ordinary least squares method. As independent variables we use a contemporaneous text sentiment value $S(t)$, a text sentiment value from day before $S(t-1)$, from 2 days before $S(t-2)$, and from 3 days before $S(t-3)$. This is done with respect to the subject company c, which is represented as an additional parameter in the equation, besides time t. We order all the companies in our dataset according to their market capitalization (total market value of all shares of the company) and divide them into ten equally sized groups. In this way we get the values for ten additional "dummy" variables dd_1 to dd_{10} (being 1 if the subject company falls into the respective group, and 0 otherwise). We include them into the regression to account for the variations of returns as a result of the company's size.

The results presented in Table 3.11 reveal that the highest estimated coefficient for the sentiment score is achieved by an SVM, but the most statistically significant result in predicting returns 1 day ahead has the proprietary Reuters NewsScope Sentiment Engine (NSE). The remaining estimated coefficients are not shown in the table and do not influence predictive power of the systems. All the company's size dummies and estimated constant are statistically significant to the level of 1%.

Table 3.12 Statistical significance of the results

	p value
$***$	$<1\%$
$**$	$<5\%$
$*$	$<10\%$
Otherwise	$\geq 10\%$

3.4.3 Visualization

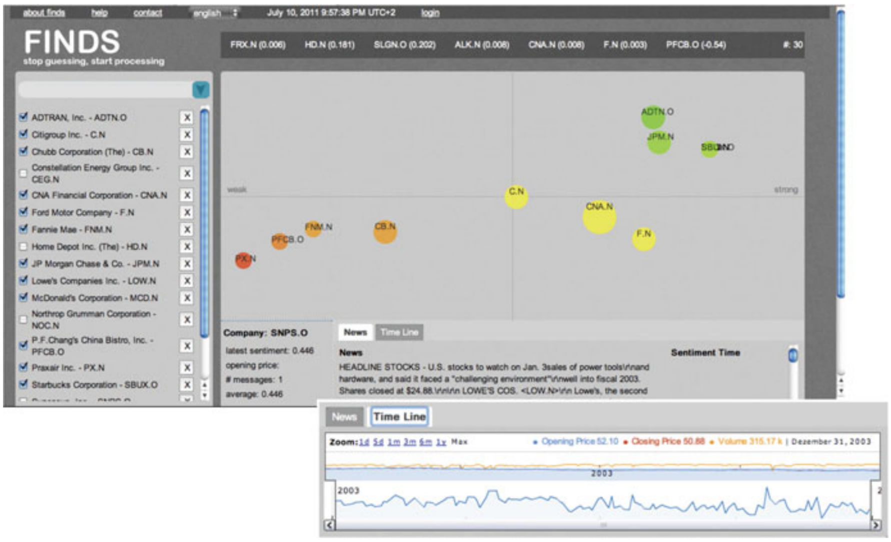
Based on our findings presented in the previous sections and on the previous works in related areas of text sentiment visualization, we developed a visualization component as a third part of the project FINDS. Visualization of data extracted from great amounts of text is a task being usually solved as a part of the wider project and often without extensive exploration of previous work in this area. All existing works cluster around two main focuses: topic detection and sentiment extraction from user opinions. For the extensive overview of other prototypes and approaches, the reader is advised to consult surveys [43, 58].

The visualization component is loosely coupled with the rest of the system via a messaging interface. This allows it to be connected and to receive inputs from various systems, either other classifier components of the FINDS system, or directly from commercially available providers of sentiment scores, like Thomson Reuters.

When a news article arrives, it is analysed by one of the FINDS classifiers, or Reuters News Score Sentiment Engine, and the sentiment score for each of the companies mentioned in the article is calculated as a real number between -1 and 1. The values near -1 indicate news that are rather bad for the affected company and implicitly signalize negative pressure on company's equity price and returns. As we learned from the previous chapters, sudden change of the sentiment are even better signals of this negative pressure, so when negative news arrives for the company that had in average positive sentiment, that is a strong signal that the downward pressure on prices will be strong in a short future period. This is why our visualization system is designed in a way to make such and similar cases clearly noticeable.

The typical screen of the FINDS visualization component is shown in Fig. 3.1. The users have a possibility to set a list of all companies they are interested in, so only these companies will be displayed. Each company is represented by the coloured circle. We can vary four dimensions for each company: x-axis position, y-axis position, size and colour.

The size of the circle representing a company and its sentiment is defined in such a manner that the visibility is kept also in the case of simultaneous representation of a great number of companies. The circle area is linearly dependent on number of messages published about a particular company, and the total area of all displayed representations is kept constant. Using this representation approach, the companies that draw more attention and are mentioned in more news stories are represented with a larger circle.



July 10, 2011 9:57:38 PM UTC+2

FRX.N (0.006) HD.N (0.181) SLGN.O (0.202) ALK.N (0.008) CNA.N (0.008) F.N (0.003) PFCB.O (-0.54) #: 30

- ✓ ADTRAN, Inc. - ADTN.O
- ✓ Citigroup Inc. - C.N
- ✓ Chubb Corporation (The) - CB.N
- ☐ Constellation Energy Group Inc. - CEG.N
- ✓ CNA Financial Corporation - CNA.N
- ✓ Ford Motor Company - F.N
- ✓ Fannie Mae - FNM.N
- ☐ Home Depot Inc. (The) - HD.N
- ✓ JP Morgan Chase & Co. - JPM.N
- ✓ Lowe's Companies Inc. - LOW.N
- ✓ McDonald's Corporation - MCD.N
- ☐ Northrop Grumman Corporation - NOC.N
- ✓ P.F.Chang's China Bistro, Inc. - PFCB.O
- ✓ Praxair Inc. - PX.N
- ✓ Starbucks Corporation - SBUX.O

Company: SNPS.O
latest sentiment: 0.446
opening price:
# messages: 1
average: 0.446

News | Sentiment Time
HEADLINE STOCKS - U.S. stocks to watch on Jan. 3sales of power tools\r\nand hardware, and said it faced a "challenging environment"\r\nwell into fiscal 2003. Shares closed at $24.88.\r\n\r\n LOWE'S COS. <LOW.N>\r\n Lowe's, the second

● Opening Price 52.10 ● Closing Price 50.88 ● Volume 315.17 k | Dezember 31, 2003

Fig. 3.1 Screenshot of the FINDS visualization component

The colour of the company's representation is chosen from the palette that ranges from green, over yellow and orange, to red. The colour directly represents the last sentiment score, by using bright red to represent sentiment value of -1 and very bad news to bright green used for the sentiment value of $+1$ and very positive news. The x position is determined in the same manner, ordering companies from left to right according to increasing recent sentiment score.

The y position of the company's representation is determined by a deviation of the recent sentiment score from the average sentiment score for the particular company, so the sentiment reversals can be easily noticed. The companies with highest sudden increase in sentiment are singled out in upper portion of display, while those companies with highest decrease in sentiment are occupying the lower portion of the display. To make this relation more obvious, we use a transformation described by (3.4) to calculate the y position of the company representation y_c, as described in [17]. $S_r(c)$ denotes deviation of current sentiment score from the average value, as described in Sect. 3.3.2. The values for constants are chosen empirically, after testing the system on realistic data:

$$y_c = C_1 \arctan(C_2 S_r(c)) + C_3, \;\; C_1 = \frac{1.056}{\pi}, \;\; C_2 = 6, \;\; C_3 = 0.5. \tag{3.4}$$

The user is offered the possibility to see the details of the selected company, like numerical values of current sentiment, average sentiment, the integral text of news story, and the historical development of stock prices for the selected company. Our goal is to offer an integrated decision support tool for the users interested in analysing influence of published news articles to market movements.

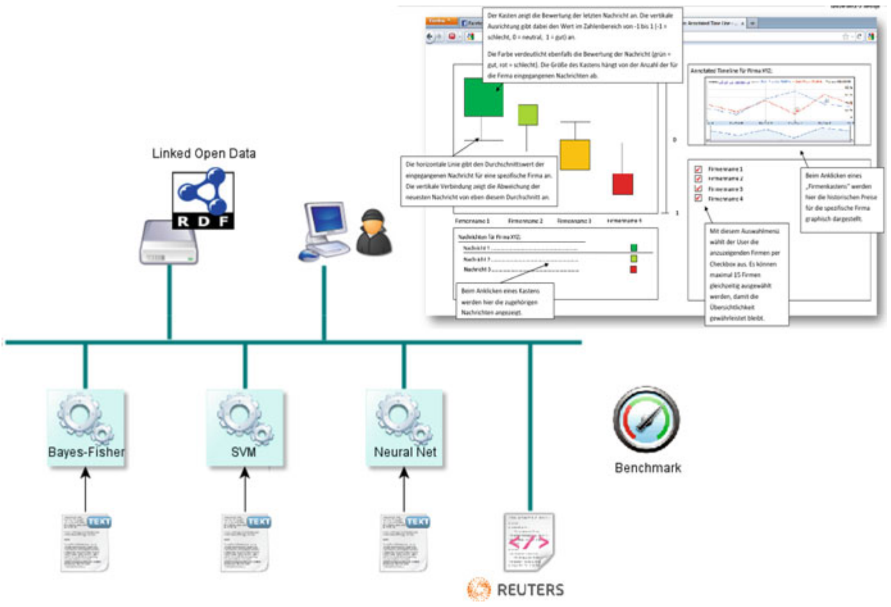

Linked Open Data
RDF
Der Kasten zeigt die Bewertung der letzten Nachricht an. Die vertikale Ausrichtung gibt dabei den Wert im Zahlenbereich von -1 bis 1 (-1 = schlecht, 0 = neutral, 1 = gut) an.
Die Farbe verdeutlicht ebenfalls die Bewertung der Nachricht (grün = gut, rot = schlecht). Die Größe des Kastens hängt von der Anzahl der für die Firma eingegangenen Nachrichten ab.
Die horizontale Linie gibt den Durchschnittswert der eingegangenen Nachricht für eine spezifische Firma an. Die vertikale Verbindung zeigt die Abweichung der neuesten Nachricht von eben diesem Durchschnitt an.
Firmenname 1
Firmenname 2
Firmenname 3
Firmenname 4
Nachrichten für Firma XYZ:
Nachricht 1
Nachricht 2
Nachricht 3
Beim Anklicken eines Kastens werden hier die zugehörigen Nachrichten angezeigt.
Annotated Timeline für Firma XYZ:
Mit diesem Auswahlmenü wählt der User die anzuzeigenden Firmen per Checkbox aus. Es können maximal 15 Firmen gleichzeitig ausgewählt werden, damit die Übersichtlichkeit gewährleistet bleibt.
Beim Anklicken eines „Firmenkastens" werden hier die historischen Preise für die spezifische Firma graphisch dargestellt.
Bayes-Fisher
SVM
Neural Net
Benchmark
TEXT
REUTERS

Fig. 3.2 Components of the FINDS system: classification engines (bottom left), benchmark component (bottom right), interface component (top left), and multi-component GUI (top right, described in detail in Sect. 3.4.3.)

3.5 Conclusion

We provided an overview of the existing text mining systems whose main purpose is predicting equity price movements on the financial markets. Some of the systems transform the input text directly to a numerical value that we call sentiment score. Others output categorical result, which in turn can be transformed into a numerical value, thus producing a sentiment score. As an example of such sentiment score properties, we show how it relates to some of the macroeconomic variables. It is suggested that this sentiment score can be transformed to reveal sentiment reversals and deviations from the average score for a company, and such transformed indicator relates better to future returns.

Finally we present the project FINDS as an integrated system that can be used for sentiment extraction from the financial news, comparison between different classification engines, and the representation of the sentiment data to the end user.

In the time when more than 50% of all trades on certain markets are executed by algorithms, as reported in [27], real time information on sentiment of published news can be used as a valuable additional indicator and input for these algorithms. We plan on building an interface between the FINDS system and other systems which can use sentiment score information. The structure of the complete system is illustrated in Fig. 3.2. In this way we could offer traders support for their buy and sell decisions, by visually filtering important news articles, but we could also support them by letting the algorithm determine the best timing and amount of the order, so the trade is executed optimally.

References

1. W. Antweiler, F.Z. Murray, Is all that talk just noise? The information content of internet stock message boards. J. Finance **59**(3), 1259–1294 (2004)
2. C. Bozic, Finds – integrative services, in *IEEE/ACS International Conference on Computer Systems and Applications, 2009 (AICCSA 2009)*, IEEE, 10–13 May 2009, pp. 61–62. doi:10.1109/AICCSA.2009.5069302
3. C. Bozic, D. Seese, Neural networks for sentiment detection in financial text, in *Proceedings of the 14th International Business Research Conference*, Melbourne, Victoria, Australia, April 2011 (World Business Institute, Australia, 2011). URL http://www.wbiconpro.com/344-Bozic.pdf
4. C. Bozic, D. Seese, News analytics and text sentiment visualization in finance, in *Proceedings of the SYMOPIS, 38. Symposium on Operational Research*, Faculty of Economics, University of Belgrade, Serbia, 2011, pp. 127–130. URL http://www.aifb.kit.edu/images/2/25/Bozic_seese_short_4pDSC.pdf
5. C. Bozic, D. Seese, News analytics: exploring predictive power of aggregated text sentiment measure, in *Proceedings of Annual Paris Conference on Money, Economy and Management*, Melbourne, Victoria, Australia, July 2011 (World Business Institute, Australia, 2011). URL http://www.wbiconpro.com/327-Caslav.pdf
6. C. Bozic, R. Riordan, D. Seese, C. Weinhardt, Towards a benchmarking framework for financial text mining, in *Information Management and Market Engineering*, ed. by D. Thomas, J. Krämer, R. Studer, C. Weinhardt. Studies on eOrganisation and Market Engineering, vol. 2 (KIT Scientific Publishing, Karlsruhe, 2010), pp. 21–36
7. R. Brown, Incorporating news into algorithmic trading strategies: increasing the signal-to-noise ratio, in *The Handbook of News Analytics in Finance*, chap. 14 (Wiley Finance, NJ, 2011), pp. 307–309
8. S.K. Chalup, A. Mitschele, Kernel methods in finance, in *Handbook on Information Technology in Finance, International Handbooks on Information Systems*, ed. by D. Seese, C. Weinhardt, F. Schlottmann (Springer, Berlin, 2008), pp. 655–687
9. V. Cho, B. Wüthrich, Combining forecasts from multiple textual data sources, in *Methodologies for Knowledge Discovery and Data Mining*, ed. by N. Zhong, L. Zhou. Lecture Notes in Computer Science, vol. 1574 (Springer, Berlin, 1999), pp. 174–179
10. V. Cho, B. Wüthrich, J. Zhang, Text processing for classification. Technical report, The Hong Kong University of Science and Technology (1998)
11. W.S.V. Cho, Knowledge discovery from distributed and textual data. PhD thesis, Department of Computer Science, Hong Kong University of Science and Technology (1999)
12. M. Costantino, Financial Information Extraction using pre-defined and user-definable Templates in the LOLITA System. PhD thesis, Laboratory for Natural Language Engineering, Department of Computer Science, University of Durham (1997)
13. M. Costantino, The LOLITA user-definable template interface. J. Comput. Inf. Tech. – CIT **9**(1), 55–69 (2001)
14. M. Costantino, R.J. Collingham, R.G. Morgan, *Natural Language Processing in Finance: Automatic Extraction of Information from Financial News Articles*. Laboratory for Natural Language Engineering, Department of Computer Science (University of Durham, UK, 1995)
15. M. Costantino, R.J. Collingham, R.G. Morgan, Financial information extraction at the University Of Durham, in *Artificial Intelligence in Accounting and Auditing: International*

Perspectives (Markus Wiener Publishers, NJ, 2005); exists also in *Proceedings of the II Meeting of Artificial Intelligence in Accounting, Finance and Tax* (University of Huelva, Spain, 1996)
16. M. Costantino, R. Morgan, R. Collingham, R. Carigliano, in *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr), 1997*. IEEE (IEEE, 1997), pp. 116–122. DOI 10.1109/CIFER.1997.618923
17. A. Darmoul, G. Hackensellner, P. Rouast, in *Findsvisions*. Abschlusspräsentation, 07 2011 (Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany)
18. S.R. Das, M.Y. Chen, Yahoo! for Amazon: Sentiment extraction from small talk on the web. Manag. Sci. **53**(9), 1375–1388 (2007)
19. P. Domingos, M. Pazzani, On the optimality of the simple Bayesian classifier under zero-one loss. Mach. Learn. **29**, 103–130 (1997)
20. J. Doornik, M. Arellano, S. Bond, *Panel Data Estimation Using DPD for OX* (Nuffield College, Oxford, 2001)
21. T. Fawcett, An introduction to ROC analysis. Pattern Recogn. Lett. **27**(8), 861–874 (2006)
22. T. Fawcett, F. Provost, Activity monitoring: Noticing interesting changes in behavior, in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '99* (ACM, New York, 1999), pp. 53–62
23. G. Fung, J. Yu, W. Lam, News sensitive stock trend prediction, in *Advances in Knowledge Discovery and Data Mining*, ed. by M.-S. Chen, P. Yu, B. Liu. Lecture Notes in Computer Science, vol. 2336 (Springer, Berlin, 2002), pp. 481–493
24. G.P.C. Fung, J. Xu Yu, W. Lam, in *2003 IEEE International Conference on Computational Intelligence for Financial Engineering, 2003*. IEEE (IEEE, 2003), pp. 395–402. DOI 10.1109/CIFER.2003.1196287
25. G. Gidófalvi, C. Elkan, Using news articles to predict stock price movements. Draft Tech Report, Department of Computer Science and Engineering, University of California, San Diego, USA (2003)
26. G. Gidófalvi, Using news articles to predict stock price movements. Project Report, Department of Computer Science and Engineering, University of California, San Diego, USA (2001)
27. T. Hendershott, R. Riordan, Algorithmic Trading and Information. Working Papers 09-08, NET Institute, March 2009. University of California, Berkeley, USA
28. S. Hitzemann, M. Uhrig-Homburg, K.-M. Ehrhart, The impact of the yearly emissions announcement on CO2 prices: an event study, in *Information Management and Market Engineering*, ed. by T. Dreier, J. Krämer, R. Studer, C. Weinhardt. Studies on eOrganisation and Market Engineering, vol. 2 (KIT Scientific Publishing, Karlsruhe, 2010), pp. 63–78
29. International Monetary Fund. World economic outlook database (2011). URL http://www.imf.org/external/ns/cs.aspx?id=28. Accessed June 12 2011
30. J. Kittrell, Sentiment reversals as buy signals, in *The Handbook of News Analytics in Finance*, chap. 9 (Wiley Finance, NJ, 2011), pp. 231–244
31. V. Lavrenko, D. Lawrie, P. Ogilvie, M. Schmill. Electronic analyst of stock behavior. Project Draft (1999). URL http://homepages.inf.ed.ac.uk/vlavrenk/doc/pitch.pdf. Center for Intelligent Information Retrieval, University of Massachusetts Amherst, USA
32. V. Lavrenko, M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen, J. Allan, in *Proceedings of the ninth international conference on Information and knowledge management* (ACM, New York, NY, USA, 2000), CIKM '00, pp. 389–396. DOI http://doi.acm.org/10.1145/354756.354845. URL http://doi.acm.org/10.1145/354756.354845
33. V. Lavrenko, M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen, J. Allan, Mining of concurrent text and time-series, in *Sixth ACMSIGKDD International Conference on Knowledge Discovery and Data Mining* (2000)
34. S.K.F. Leung, Automatic stock market predictions from world wide web data. Master's thesis, Department of Computer Science, Hong Kong University of Science and Technology (1997)
35. X. Liang, Impacts of internet stock news on stock markets based on neural networks, in *Advances in Neural Networks? ISNN 2005*, ed. by J. Wang, X.F. Liao, Z. Yi, Lecture Notes in Computer Science, vol. 3497 (Springer, Berlin, 2005), pp. 811–811

36. X. Liang, R.C. Chen, in *Proceedings of ICNN&B '05. International Conference on Neural Networks and Brain, 2005*, vol. 2, IEEE, 2005, pp. 893 –898. doi:10.1109/ICNNB.2005.1614765
37. U. Lösch. Event and sentiment detection in financial markets (2008). URL http://www.aifb.kit.edu/images/b/bc/2008_1774_L%C3%B6sch_Event_and_Senti_1.pdf
38. U. Lösch, R. Studer, C. Weinhardt, New event detection in financial news analysis, in *Information Management and Market Engineering*, ed. by T. Dreier, J. Krämer, R. Studer, C. Weinhardt. Studies on eOrganisation and Market Engineering, vol. 2 (KIT Scientific Publishing, Karlsruhe, 2010), pp. 3–20
39. M.A. Mittermayer, in *Proceedings of the 37th Annual Hawaii International Conference on System Sciences, 2004.* (IEEE Computer Society Press, 2004). DOI 10.1109/HICSS.2004.1265201
40. M.-A. Mittermayer, Einsatz von Text Mining zur Prognose kurzfristiger Trends von Aktienkursen nach der Publikation von Unternehmensnachrichten. PhD thesis, Uniersitt Bern (2006)
41. M.-A. Mittermayer, G.F. Knolmayer, NewsCATS: a news categorization and trading system, in *IEEE International Conference on Data Mining*, pp. 1002–1007 (2006)
42. R. Morgan, R. Garigliano, P. Callaghan, S. Poria, M. Smith, C. Cooper, University of Durham: Description of the LOLITA system as used in MUC-6, in *Proceedings of the 6th conference on Message understanding, MUC6 '95* (Association for Computational Linguistics, Stroudsburg, 1995), pp. 71–85
43. B. Pang, L. Lee, Opinion mining and sentiment analysis. Found. Trends Inf. Retr. **2**, 1–135 (2008)
44. D. Peramunetilleke, R.K. Wong, Currency exchange rate forecasting from news headlines, in *ADC'02: Proceedings of the 13th Australasian Database Conference*, Darlinghurst, Australia (Australian Computer Society, Australia, 2002), pp. 131–139
45. J.C. Platt, Fast training of support vector machines using sequential minimal optimization, in *Advances in Kernel Methods*, ed. by B. Schölkopf, C.J.C. Burges, A.J. Smola (MIT, Cambridge, 1999), pp. 185–208
46. F.J. Provost, T. Fawcett, R. Kohavi, The case against accuracy estimation for comparing induction algorithms, in *Proceedings of the Fifteenth International Conference on Machine Learning, ICML '98*, San Francisco, CA, USA (Morgan Kaufmann Publishers, MA, 1998), pp. 445–453
47. R. Riordan, A. Storkenmaier, M. Wagener, Public Information Arrival: Price Discovery and Liquidity in Electronic Limit Order Markets. Working Paper Series, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany (2011)
48. A. Schulz, M. Spiliopoulou, K. Winkler, Kursrelevanzprognose von ad-hoc-meldungen: Text mining wider die informationsüberlastung im mobile banking. Wirtschaftsinformatik **2**, 181–200 (2003)
49. R.P. Schumaker, H. Chen, in *Proceedings of the 12th Americas Conference on Information Systems (AMCIS-2006)* (AIS, Atlanta, GA, USA, 2006). URL http://www.robschumaker.com/publications/AMCIS%20\discretionary-%20Textual%20Analysis%20of%20Stock%20Market%20Prediction%20Using%20Financial%20News%20Articles.pdf
50. R.P. Schumaker, Analyzing representational schemes of financial news articles, in *The Third China Summer Workshop on Information Systems*, 2009
51. R.P. Schumaker. Analyzing parts of speech and their impact on stock price. Conference paper (2010). URL http://www.robschumaker.com/publications/IIMA%20\discretionary-%20A%20Discrete%20Stock%20Price%20Prediction%20Engine%20Based%20on%20Financial%20News.pdf
52. R.P. Schumaker, Analyzing parts of speech and their impact on stock price. Comm. Int. Inform. Manag. Assoc. **10**, 1–10 (2010)
53. R.P. Schumaker, An analysis of verbs in financial news articles and their impact on stock price, in *NAACL Workshop on Social Media and Computational Linguistics* (2010)
54. R.P. Schumaker, H. Chen, A quantitative stock prediction system based on financial news. Inform. Process. Manag. **45**, 571–583 (2009)

55. R.P. Schumaker, H. Chen, Textual analysis of stock market prediction using breaking financial news: The AZFINTEXT system. Assoc. Comput. Mach. Trans. Inform. Syst. **27**(2), 12:1–12:19 (2009)
56. R.P. Schumaker, H. Chen, A discrete stock price prediction engine based on financial news. IEEE Comp. **43**, 51–56 (2010)
57. R.P. Schumaker, Y. Zhang, C.N. Huang. Sentiment analysis of financial news articles. conference paper (2009). URL http://www.robschumaker.com/publications/IIMA%20\discretionary-%20Sentiment%20Analysis%20of%20Financial%20News%20Articles.pdf
58. A. Šilić, B. Bašić, Visualization of text streams: A survey, in *Knowledge-Based and Intelligent Information and Engineering Systems*, ed. by R. Setchi, I. Jordanov, R. Howlett, L. Jain. Lecture Notes in Computer Science, vol. 6277 (Springer, Berlin, 2010), pp. 31–43
59. A. Storkenmaier, M. Wagener, C. Weinhardt, Public Information in Fragmented Markets. Working Paper Series, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany (2011)
60. P.C. Tetlock, Giving content to investor sentiment: The role of media in the stock market. J. Finance **62**(3), 1139–1168 (2007)
61. P.C. Tetlock, M. Saar-Tsechansky, S. Macskassy, More than words: Quantifying language to measure firms' fundamentals. J. Finance **63**(3), 1437–1467 (2008)
62. J.D. Thomas, News and trading rules. PhD thesis, Carnegie Mellon University (2003)
63. J. Thorsten, *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms* (Kluwer, Norwell, 2002)
64. K.M. Tolle, H. Chen, Comparing noun phrasing techniques for use with medical digital library tools. J. Am. Soc. Inform. Sci. **51**(4), 352–370 (2000)
65. C. Ullrich, D. Seese, S. Chalup, Foreign exchange trading with support vector machines, in *Advances in Data Analysis, Studies in Classification, Data Analysis, and Knowledge Organization*, ed. by R. Decker, H.-J. Lenz (Springer, Berlin, 2007), pp. 539–546
66. B. Wüthrich, D. Permunetilleke, S. Leung, V. Cho, J. Zhang, W. Lam. Daily prediction of major stock indices from textual www data. Poster (1998). URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.2979
67. S.S. Zhang, C. Weinhardt, R. Riordan, Market responses to the introduction of interactive data: the case of XBRL, in *Information Management and Market Engineering*, ed. by T. Dreier, J. Krämer, R. Studer, C. Weinhardt. Studies on eOrganisation and Market Engineering, vol. 2 (KIT Scientific Publishing, Karlsruhe, 2010), pp. 55–62

Chapter 4
Modelling and Trading the Greek Stock Market with Hybrid ARMA-Neural Network Models

Christian L. Dunis, Jason Laws, and Andreas Karathanasopoulos

Abstract The motivation for this chapter is to investigate the use of alternative novel neural network architectures when applied to the task of forecasting and trading the ASE 20 Greek Index using only autoregressive terms as inputs. This is done by benchmarking the forecasting performance of six different neural network designs representing a Higher Order Neural Network (HONN), a Recurrent Network (RNN), a classic Multilayer Perceptron (MLP), a Hybrid Higher Order Neural Network, a Hybrid Recurrent Neural Network and a Hybrid Multilayer Perceptron Neural Network with some traditional techniques, either statistical such as an autoregressive moving average model (ARMA) or technical such as a moving average convergence/divergence model (MACD), plus a naïve trading strategy. More specifically, the trading performance of all models is investigated in a forecast and trading simulation on ASE 20 fixing time series over the period 2001–2008 using the last one and a half year for out-of-sample testing. We use the ASE 20 daily fixing as many financial institutions are ready to trade at this level and it is therefore possible to leave orders with a bank for business to be transacted on that basis. As it turns out, the hybrid-HONNs do remarkably well and outperform all other models in a simple trading simulation exercise. However, when more sophisticated trading strategies using confirmation filters and leverage are applied, the hybrid-HONN network produces better results and outperforms all other neural network and traditional statistical models in terms of annualised return.

C.L. Dunis (✉)
Liverpool John Moores University, Liverpool, UK
e-mail: C.Dunis@ljmu.ac.uk

J. Laws
University of Liverpool Management School, Liverpool, UK
e-mail: J.Laws@liverpool.ac.uk

A. Karathanasopoulos
London Metropolitan University, London, UK
e-mail: a.karathanasopoulos@londonmet.ac.uk

4.1 Introduction

The use of intelligent systems for market predictions has been widely established. This chapter deals with the application of hybridised computing techniques for forecasting the Greek stock market. The development of accurate techniques is critical to economists, investors and analysts. This task is getting more and more complex as financial markets are getting increasingly interconnected and interdependent. The traditional statistical methods, on which forecasters were reliant in recent years, seem to fail to capture the interrelationship between market variables. This chapter investigates methods capable of identifying and capturing all the discontinuities, the non-linearities and the high-frequency multipolynomial components characterising the financial series today. A model category that promises such effective results is the combination of autoregressive models such as ARMA model with neural networks named hybrid-neural network model. Many researchers have argued that combining several models for forecasting gives better estimates by taking advantage of each model's capabilities when comparing them with single time series models.

The motivation for this chapter is to investigate the use of several new neural networks techniques combined with ARMA model in order to overcome these limitations using autoregressive terms as inputs. This is done by benchmarking six different neural network architectures representing a Multilayer Perceptron (MLP), a Higher Order Neural Network (HONN), a Recurrent Neural Network (RNN), a Hybrid Higher Order Neural Network, a Hybrid Recurrent Neural Network and a Hybrid Multilayer Perceptron Neural Network. Their trading performance on the ASE 20 time series is investigated and is compared with some traditional statistical or technical methods such as an autoregressive moving average (ARMA) model or a moving average convergence/divergence (MACD) model, and a naïve trading strategy.

As it turns out, the hybrid-HONN demonstrates a remarkable performance and outperforms all other models in a simple trading simulation exercise. On the other hand, when more sophisticated trading strategies using confirmation filters and leverage are applied, HONNs outperform all models in terms of annualised return. Our conclusion corroborates those of Lindemann et al. [25] and Dunis et al. [11] where HONNs also demonstrate a forecasting superiority on the EUR/USD series over more traditional techniques such as an MACD and a naïve strategy. However, the RNN, which performed remarkably well, shows a disappointing performance in this research: this may be due to their inability to provide good enough results when only autoregressive terms are used as inputs.

The rest of the chapter is organised as follows. In Sect. 4.2, we present the literature relevant to the hybrid-neural networks, the RNN, the HONNs and the multilayer perceptron. Section 4.3 describes the data set used for this research and its characteristics. An overview of the different neural network models and statistical techniques is given in Sect. 4.4. Section 4.5 gives the empirical results of all the models considered and investigates the possibility of improving their performance with the application of more sophisticated trading strategies. Section 4.6 provides some concluding remarks.

4.2 Literature Review

The motivation for this chapter is to apply some of the most promising new neural network architectures combining them with autoregressive models (in our case ARMA model) which have been developed recently with the purpose to overcome the numerous limitations of the more classic neural architectures and to assess whether they can achieve a higher performance in a trading simulation using only autoregressive series as inputs.

Combining different models can increase the chance to capture different patterns in the data and improve forecasting performance. Several empirical studies have already suggested that by combining several different models, forecasting accuracy can often be improved over an individual model. Using hybrid models or combining several models has become a common practice to improve the forecasting accuracy since the well-known M-competition [27] in which combinations of forecasts from more than one model often led to improved forecasting performance. The basic idea of the model combination in forecasting is to use each model's unique feature to capture different patterns in the data. Both theoretical and empirical findings suggest that combining different methods can be an effective and efficient way to improve forecasts [26, 28, 29, 41]. Research in time series forecasting argues that predictive performance improves the combined models [2, 5, 17, 18, 35, 38, 40, 42].

The reason for combining models comes from the assumption that either one cannot identify the true data-generating process [35] or that a single model may not be sufficient to identify all the characteristics of the time series [40]. Moreover, hybrid-neural networks have not been used until the moment that scientists started to investigate not only the benefits of hybrid-neural networks against other statistical methods but also the differences between different combinations of hybrid-neural networks with other statistical models following hybrid GARCH-NN approach [39] and hybrid ARIMA/ ARCH-NN [14].

RNNs have an activation feedback which embodies short-term memory allowing them to learn extremely complex temporal patterns. Their superiority against feedfoward networks when performing non-linear time series prediction is well documented in [1, 6]. In financial applications, Kamijo et al. [21] applied them successfully to the recognition of stock patterns of the Tokyo stock exchange while Tenti [34] achieved remarkable results using RNNs to forecast the exchange rate of the Deutsche Mark. Tino et al. [37] use them to trade successfully the volatility of the DAX and the FTSE 100 using straddles while Dunis and Huang [8], using continuous implied volatility data from the currency options market, obtain remarkable results for their GBP/USD and USD/JPY exchange rate volatility trading simulation.

HONNs were first introduced by introduced by Giles and Maxwell [16] as a fast learning network with increased learning capabilities. Although their function approximation superiority over the more traditional architectures is well documented in the literature (see among others [24, 31, 32]), their use in finance so far has been limited. This has changed when scientists started to investigate not only the benefits of neural networks (NNs) against the more traditional statistical techniques but also the differences between the different NN model architectures. Practical applications have now verified the theoretical advantages of HONNs by demonstrating their superior forecasting ability and put them in the front line of research in financial forecasting. For example, Dunis et al. [9] use them to forecast successfully the gasoline crack spread while Fultcher et al. [15] apply HONNs to forecast the AUD/USD exchange rate, achieving a 90% accuracy. However, Dunis et al. [10] show that, in the case of the futures spreads and for the period under review, the MLPs performed better compared with HONNs and RNNs. Moreover, Dunis et al. [12], who also study the EUR/USD series for a period of 10 years, demonstrate that when multivariate series are used as inputs the HONNs, RNN and MLP networks have a similar forecasting power. Finally, Dunis et al. [11], in a paper with a methodology identical to that used in this research, demonstrate that HONN and the MLP networks are superior in forecasting the EUR/USD ECB fixing until the end of 2007, compared to the RNN networks, an ARMA model, a MACD and a naïve strategy.

4.3 The ASE-20 Greek and Related Financial Data

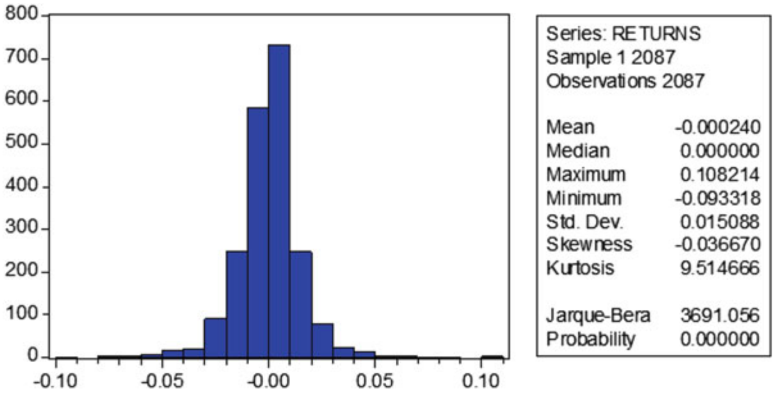
For futures on the FTSE/ASE-20 that are traded in derivatives markets the underlying asset is the blue chip index FTSE/ASE-20. The FTSE/ASE-20 index is based on the 20 largest ASE stocks. It was developed in 1997 by the partnership of ASE with FTSE International and is already an established benchmark. It represents over 50% of ASE's total capitalisation and currently has a heavier weight on banking, telecommunication and energy stocks.

The futures contract on the index FTSE/ASE-20 is cash settled in the sense that the difference between the traded price of the contract and the closing price of the index on the expiration day of the contract is settled between the counterparties in cash. As a matter of fact, as the price of the contract changes daily, it is cash settled on a daily basis, up until the expiration of the contract. The futures contract is traded in index points, while the monetary value of the contract is calculated by multiplying the futures price by the multiplier 5 EUR per point. For example, a contract trading at 1,400 points has a value of 7,000 EUR.

The ASE 20 futures is therefore a tradable level which makes our application more realistic and this is the series that we investigate in this chapter.[1]

The observed ASE 20 time series is non-normal (Jarque-Bera statistics confirms this at the 99% confidence interval) containing slight skewness and high kurtosis.

[1]We examine the ASE 20 since its first trading day on 21 January 2001, and until 31 December 2008, using the continuous data available from datastream.



Series: RETURNS
Sample 1 2087
Observations 2087

Mean	-0.000240
Median	0.000000
Maximum	0.108214
Minimum	-0.093318
Std. Dev.	0.015088
Skewness	-0.036670
Kurtosis	9.514666
Jarque-Bera	3691.056
Probability	0.000000

Fig. 4.1 ASE 20 returns summary statistics (total data set)

Table 4.1 The ASE 20 dataset

Name of period	Trading days	Beginning	End
Total data set	2,087	21 January 2001	31 December 2008
Training data set	1,719	29 January 2001	30 August 2007
Out-of-sample data set (validation set)	349	31 August 2007	31 December 2008

It is also non-stationary and we decided to transform the ASE 20 series into stationary series of rates of return.[2] A distribution of returns can be seen in Fig. 4.1.

Given the price level $P_1, P_2, \ldots, P_t$, the rate of return at time t is formed by

$$R_t = \left(\frac{P_t}{P_{t-1}}\right) - 1. \tag{4.1}$$

As inputs to our networks and based on the autocorrelation function and some ARMA experiments we selected two sets of autoregressive and moving average terms of the ASE 20 returns and the 1-day Riskmetrics volatility series.

In order to train the neural networks we further divided our data set as shown in 4.2. A division of the data set can be seen in Tables 4.1 and 4.2 while a graphical representation of the total data set can be seen in Fig. 4.2.

Furthermore, the inputs and lag structure used for the neural network models can be seen in Table 4.3. Each of the inputs and lags were optimised in sample according to annualised returns (Table 4.4).

[2]Confirmation of its stationary property is obtained at the 1% significance level by both the Augmented Dickey Fuller (ADF) and Phillips-Perron (PP) test statistics.

Table 4.2 The neural networks data sets

Name of period	Trading days	Beginning	End
Total data set	2,087	21 January 2001	31 December 2008
Training data set	1,373	29 January 2001	03 May 2006
Test data set	346	04 May 2006	30 August 2007
Out-of-sample data set (validation set)	349	31 August 2007	31 December 2008

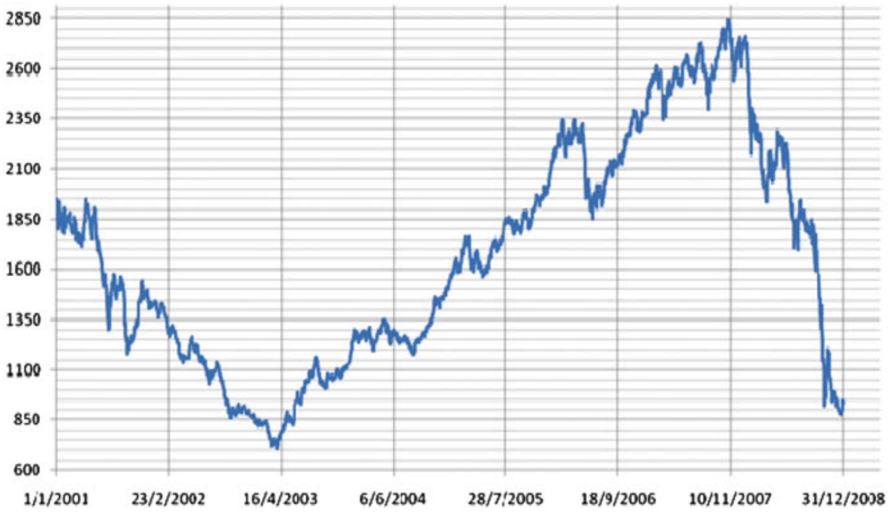


Fig. 4.2 ASE 20 fixing prices (total data set)

Table 4.3 Explanatory variables for traditional neural networks

Number	Variable	Lag
1	Athens Composite all share return	1
2	Athens Composite all share return	3
3	Athens Composite all share return	6
4	Athens Composite all share return	8
5	Athens Composite all share return	10
6	Athens Composite all share return	13
7	Athens Composite all share return	14
8	Moving average of the Athens Composite all share return	15
9	Athens Composite all share return	16
10	Athens Composite all share return	18
11	Moving average of the Athens Composite all share return	19

Table 4.4 Explanatory variables for hybrid-neural networks

Number	Variable	Lag
1	Athens Composite all share return	1
2	Athens Composite all share return	3
3	Athens Composite all share return	5
4	Athens Composite all share return	7
5	Athens Composite all share return	8
6	Athens Composite all share return	9
7	Athens Composite all share return	12
8	Athens Composite all share return	13
9	Moving average of the Athens Composite all share return	14
10	Athens Composite all share return	15
11	Athens Composite all share return	16
12	Moving average of the Athens Composite all share return	17
13	1-day Riskmetrics volatility	1

4.4 Forecasting Models

4.4.1 Benchmark Models

In this chapter, we benchmark our neural network models with three traditional strategies, namely an autoregressive moving average model (ARMA), a moving average convergence/divergence technical model (MACD) and a naïve strategy.

4.4.1.1 Naïve Strategy

The naïve strategy simply takes the most recent period change as the best prediction of the future change. The model is defined by

$$\hat{Y}_{t+1} = Y_t, \tag{4.2}$$

where Y_t is the actual rate of return at period t and $\hat{Y}_{t+1}$ is the forecast rate of return for the next period.

The performance of the strategy is evaluated in terms of trading performance via a simulated trading strategy.

4.4.1.2 Moving Average

The moving average model is defined as

$$M_t = \frac{Y_t + Y_{t-1} + Y_{t-2} + \cdots + Y_{t-n+1}}{n}, \tag{4.3}$$

where M_t is the moving average at time t, n is the number of terms in the moving average and Y_t is the actual rate of return at period t.

The MACD strategy used is quite simple. Two moving average series are created with different moving average lengths. The decision rule for taking positions in the market is straightforward. Positions are taken if the moving averages intersect. If the short-term moving average intersects the long-term moving average from below a "long" position is taken. Conversely, if the long-term moving average is intersected from above a "short" position is taken.[3]

The forecaster must use judgement when determining the number of periods n on which to base the moving averages. The combination that performed best over the in-sample sub-period was retained for out-of-sample evaluation. The model selected was a combination of the ASE 20 and its 7-day moving average, namely $n = 1$ and 7, respectively, or a (1, 7) combination. The performance of this strategy is evaluated solely in terms of trading performance.

4.4.1.3 ARMA Model

Autoregressive moving average models (ARMA) assume that the value of a time series depends on its previous values (the autoregressive component) and on previous residual values (the moving average component).[4]

The ARMA model takes the form

$$Y_t = \varphi_0 + \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \cdots + \varphi_p Y_{t-p} + \varepsilon_t - w_1 \varepsilon_{t-1} - w_2 \varepsilon_{t-2} - \cdots - w_q \varepsilon_{t-q}, \tag{4.4}$$

where:

- Y_t is the dependent variable at time t
- $Y_{t-1}, Y_{t-2}, \ldots, Y_{t-p}$ are the lagged dependent variables
- $\varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_p$ are regression coefficients
- ε_t is the residual term
- $\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-p}$ are previous values of the residual
- $w_1, w_2, \ldots, w_q$ are weights

Using as a guide the correlogram in the training and the test sub-periods we have chosen a restricted ARMA (7, 7) model. All of its coefficients are significant at the 99% confidence interval. The null hypothesis that all coefficients (except the constant) are not significantly different from zero is rejected at the 99% confidence interval (see section "ARMA Model" in Appendix).

[3] A "long" ASE 20 position means buying the index at the current price, while a "short" position means selling the index at the current price.

[4] For a full discussion on the procedure, refer to [3, 30].

The selected ARMA model takes the form

$$\begin{aligned} Y_t = 2.90 \cdot 10^{-4} + 0.376 Y_{t-1} - 0.245 Y_{t-3} - 0.679 Y_{t-7} \\ + 0.374 \varepsilon_{t-1} - 0.270 \varepsilon_{t-3} - 0.677 \varepsilon_{t-7}. \end{aligned} \tag{4.5}$$

The model selected was retained for out-of-sample estimation. The performance of the strategy is evaluated in terms of traditional forecasting accuracy and in terms of trading performance (statistical measures are given in Sect. 4.4.2.5).

4.4.2 Neural Networks and Hybrid-Neural Networks

Neural networks exist in several forms in the literature. The most popular architecture is the multilayer perceptron (MLP).

A standard neural network has at least three layers. The first layer is called the input layer (the number of its nodes corresponds to the number of explanatory variables). The last layer is called the output layer (the number of its nodes corresponds to the number of response variables). An intermediary layer of nodes, the hidden layer, separates the input from the output layer. Its number of nodes defines the amount of complexity the model is capable of fitting. In addition, the input and hidden layer contain an extra node, called the bias node. This node has a fixed value of one and has the same function as the intercept in traditional regression models. Normally, each node of one layer has connections to all the other nodes of the next layer.

The network processes information as follows: the input nodes contain the value of the explanatory variables. Since each node connection represents a weight factor, the information reaches a single hidden layer node as the weighted sum of its inputs. Each node of the hidden layer passes the information through a non-linear activation function and passes it on to the output layer if the calculated value is above a threshold.

The training of the network (which is the adjustment of its weights in the way that the network maps the input value of the training data to the corresponding output value) starts with randomly chosen weights and proceeds by applying a learning algorithm called backpropagation of errors [33].[5] The learning algorithm simply tries to find those weights which minimise an error function (normally the sum of all squared differences between target and actual values). Since networks with sufficient hidden nodes are able to learn the training data (as well as their outliers and their noise) by heart, it is crucial to stop the training procedure at the right time to prevent overfitting (this is called "early stopping"). This can

[5]Backpropagation networks are the most common multilayer networks and are the most commonly used type in financial time series forecasting [20].

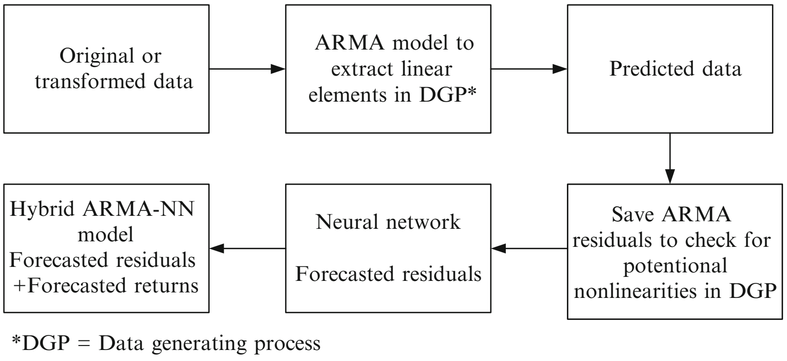

Original or transformed data
ARMA model to extract linear elements in DGP*
Predicted data
Save ARMA residuals to check for potential nonlinearities in DGP
Neural network
Forecasted residuals
Hybrid ARMA-NN model
Forecasted residuals +Forecasted returns
*DGP = Data generating process

Fig. 4.3 The architecture of hybrid ARMA-neural network model

be achieved by dividing the data set into three subsets, respectively, called the training and test sets used for simulating the data currently available to fit and tune the model and the validation set used for simulating future values. The network parameters are then estimated by fitting the training data using the above-mentioned iterative procedure (backpropagation of errors). The iteration length is optimised by maximising the forecasting accuracy for the test data set. Our networks, which are specially designed for financial purposes, will stop training when the profit of our forecasts in the test sub-period is maximised. Then the predictive value of the model is evaluated applying it to the validation data set (out-of-sample data set).

There is a range of combination techniques that can be applied to forecasting the attempt to overcome some deficiencies of single models. The combining method aims at reducing the risk of using an inappropriate model by combining several to reduce the risk of failure. Typically this is done because the underlying process cannot easily be determined [19].

Combining methods involves using several redundant models designed for the same function, where the diversity of the components is thought important [4]. The procedure of making a hybrid forecasting time series model can be achieved by combining an ARMA process in order to learn the linear component of the conditional mean pattern with an artificial neural network process designed to learn its non-linear elements. The construction of the hybrid ARMA-neural network model in details is in Figs. 4.3 and 4.7.

4.4.2.1 The Multilayer Perceptron Model Architecture

The network architecture of a "standard" MLP looks as presented in Fig. 4.4 (the bias nodes are not shown here for the sake of simplicity).
where:

- $x_t^{[n]}$ $(n = 1, 2, \ldots, k+1)$ are the model inputs (including the input bias node) at time t

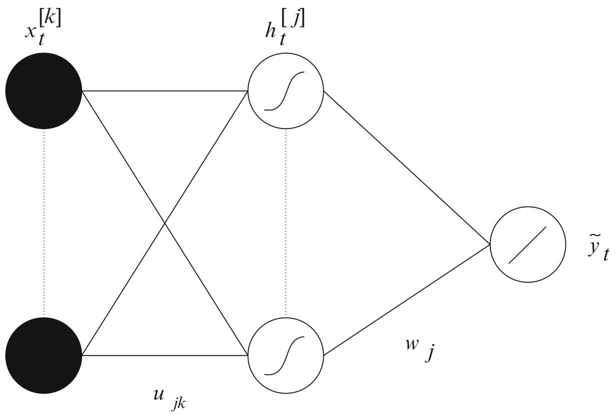


Fig. 4.4 A single output, fully connected MLP model

- $h_t^{[m]}$ $(m = 1,2,\ldots,j+1)$ are the hidden nodes outputs (including the hidden bias node)
- $\tilde{y}_t$ is the MLP model output
- u_{jk} and w_j are the network weights
- is the sigmoid transfer function

$$S(x) = \frac{1}{1+e^{-x}} \tag{4.6}$$

- is a linear function

$$F(x) = \sum_i x_i. \tag{4.7}$$

The error function to be minimised is

$$E\left(u_{jk}, w_j\right) = \frac{1}{T}\sum_{t=1}^{T}\left[y_t - \tilde{y}_t\left(u_{jk}, w_j\right)\right]^2 \tag{4.8}$$

with y_t being the target value.

4.4.2.2 The Recurrent Network Architecture

Our next model is the RNN. While a complete explanation of RNN models is beyond the scope of this chapter, we present below a brief explanation of the significant differences between RNN and MLP architectures. For an exact specification of the recurrent network, see [13].

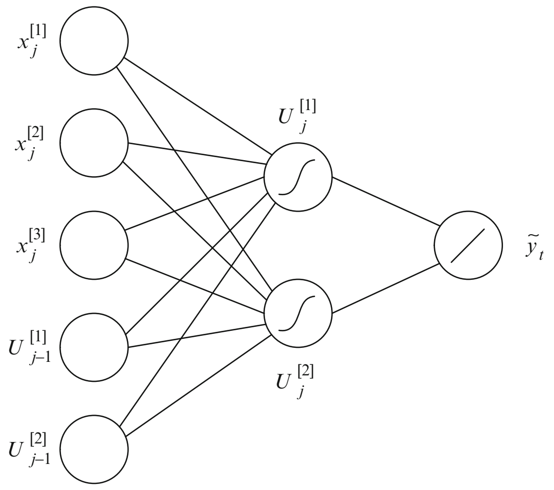


Fig. 4.5 Elman recurrent neural network architecture with two nodes on the hidden layer

A simple recurrent network has activation feedback, which embodies short-term memory. The advantages of using recurrent networks over feedforward networks, for modelling non-linear time series, has been well documented in the past. However as described in Tenti [34] "the main disadvantage of RNNs is that they require substantially more connections, and more memory in simulation, than standard backpropagation networks," thus resulting in a substantial increase in computational time. However having said this RNNs can yield better results in comparison to simple MLPs due to the additional memory inputs.

A simple illustration of the architecture of an Elman RNN is presented in Fig. 4.5. where:

- $x_t^{[n]}$ $(n = 1, 2, \ldots, k+1)$, $u_t^{[1]}$, $u_t^{[2]}$ are the model inputs (including the input bias node) at time t
- $\tilde{y}_t$) is the recurrent model output
- $d_t^{[f]}$ $(f = 1, 2)$ and $w_t^{[n]}$ $(n = 1, 2, \ldots, k+1)$ are the network weights
- $U_t^{[f]}$ $(f = 1, 2)$ is the output of the hidden nodes at time t
- is the sigmoid transfer function (4.6)
- is the linear function (4.7)

The error function to be minimised is

$$E\left(d_t, w_t\right) = \frac{1}{T}\sum_{t=1}^{T}\left[y_t - \tilde{y}_t\left(d_t, w_t\right)\right]^2. \tag{4.9}$$

In short, the RNN architecture can provide more accurate outputs because the inputs are (potentially) taken from all previous values (see inputs $U_{j-1}^{[1]}$ and $U_{j-1}^{[2]}$ in Fig. 4.5).

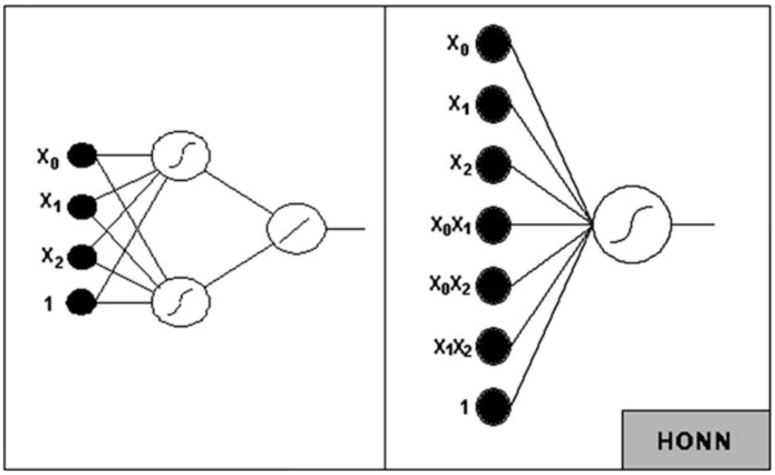


Fig. 4.6 *Left*, MLP with three inputs and two hidden nodes; *right*, second-order HONN with three inputs

4.4.2.3 The Higher Order Neural Network Architecture

HONNs were first introduced by Giles and Maxwell [16] and were called “Tensor Networks.” Although the extent of their use in finance has so far been limited, Knowles et al. [23] show that, with shorter computational times and limited input variables, “the best HONN models show a profit increase over the MLP of around 8%” on the EUR/USD time series. For Zhang et al. (2002), a significant advantage of HONNs is that HONN models are able to provide some rationale for the simulations they produce and thus can be regarded as “open box” rather than “black box.” HONNs are able to simulate higher frequency, higher order non-linear data and consequently provide superior simulations compared to those produced by artificial neural networks (ANNs). Furthermore HONNs clearly outperform in terms of annualised return and this enables Dunis et al. [11] to conclude with confidence over their forecasting superiority and their stability and robustness through time.

While they have already experienced some success in the field of pattern recognition and associative recall,[6] HONNs have only started recently to be used in finance. The architecture of a three-input second-order HONN is shown in Fig. 4.6. where:

- $x_t^{[n]}$ $(n = 1, 2, \ldots, k+1)$, $u_t^{[1]}$, $u_t^{[2]}$ are the model inputs (including the input bias node) at time t
- $\tilde{y}_t$ is the HONNs model output
- u_{jk} are the network weights

[6]Associative recall is the act of associating two seemingly unrelated entities, such as smell and colour. For more information see [22].

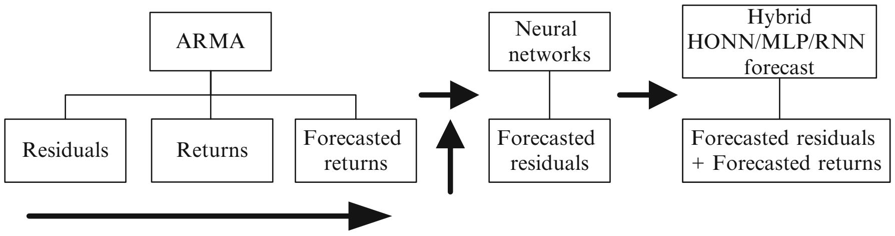


Fig. 4.7 The architecture of hybrid ARMA-neural network model

- ● are the model inputs
- is the sigmoid transfer function (4.6)
- is the linear function (4.7)

The error function to be minimised is

$$E\left(u_{jk}, w_j\right) = \frac{1}{T}\sum_{t=1}^{T}\left[y_t - \tilde{y}_t\left(u_{jk},\right)\right]^2 \tag{4.10}$$

with y_t being the target value.

HONNs use joint activation functions; this technique reduces the need to establish the relationships between inputs when training. Furthermore this reduces the number of free weights and means that HONNS are faster to train than even MLPs. However because the number of inputs can be very large for higher order architectures, orders of 4 and over are rarely used.

Another advantage of the reduction of free weights means that the problems of overfitting and local optima affecting the results of neural networks can be largely avoided. For a complete description of HONNs see [23].

4.4.2.4 The Hybrid HONN, MLP, RNN Architecture

The methodology we follow to construct the hybrid ARMA-neural network is divided into three steps. In a first step we take the residuals from the ARMA model. In a second step, we forecast the residuals with our neural network model. In a third step, we create the hybrid model by adding the forecasted returns from the ARMA model with the forecasted residuals from the second step. The architecture of the neural hybrid models can be seen in Fig. 4.7.

4.4.2.5 Forecasting Accuracy Measures

As it is standard in the literature, in order to evaluate statistically our forecasts, the RMSE, the MAE, the MAPE and the Theil-U statistics are computed. The

Table 4.5 Out-of-sample statistical performance

	RMSE	MAE	MAPE (%)	THEIL-U
NAIVE	0.0329	0.0234	811.13	0.6863
MACD	0.0254	0.0174	393.44	0.7534
ARMA	0.0239	0.0161	115.00	0.9446
MLP	0.0470	0.0163	106.97	0.9661
RNN	0.0241	0.0170	275.23	0.8287
HONN	0.0240	0.0299	679.96	0.7289
Hybrid-MLP	0.0238	0.0160	113.19	0.8891
Hybrid-RNN	0.0237	0.0160	112.83	0.8873
Hybrid-HONN	0.0237	0.0159	113.00	0.8868

RMSE and MAE statistics are scale-dependent measures but give a basis to compare volatility forecasts with the realised volatility while the MAPE and the Theil-U statistics are independent of the scale of the variables. In particular, the Theil-U statistic is constructed in such a way that it necessarily lies between zero and one, with zero indicating a perfect fit. A more detailed description of these measures can be found on [7, 30, 36] while their mathematical formulas are in section "Performance Measures" in Appendix. For all four of the error statistics retained (RMSE, MAE, MAPE and Theil-U) the lower the output, the better the forecasting accuracy of the model concerned. In Table 4.5 we present our results for the out-of-sample period.

As can be seen from Table 4.5 and section "Statistical Results in the Training and Test Sub-periods" in the Appendix for the in-sample period, hybrid-HONNs outperform all other models and present the most accurate forecasts in statistical terms in both in- and out-of-sample periods. It seems that their ability to capture higher order correlations gave them a considerable advantage compared to the other models. Hybrid-RNNs come second and hybrid-MLPs come third in our statistical evaluation in both periods. Furthermore, it is worth noting that the time that we need to train our HONNs was less than the time needed for the RNNs and the MLPs.

4.4.2.6 Empirical Trading Simulation Results

The trading performance of all the models considered in the validation subset is presented in the Table 4.6. We choose the network with the higher profit in the test sub-period. Our trading strategy applied is simple and identical for all the models: go or stay long when the forecast return is above zero and go or stay short when the forecast return is below zero. Section "Empirical Results in the Training and Test Sub-periods" in Appendix provides the performance of all the NNs in the training and the test sub-periods while sections "Performance Measures" and "Networks Characteristics" in Appendix provide the characteristics of our networks and the

Table 4.6 Trading performance results

	Information ratio[a]	Annualised volatility[a] (%)	Annualised return[a] (%)	Maximum drawdown[a] (%)	Positions taken[b]
NAIVE	0.32	36.70	11.42	−49.41	119
MACD	0.46	38.12	17.63	−50.63	38
ARMA	0.20	38.13	7.68	−36.50	72
MLP	0.60	38.11	22.99	−36.26	105
RNN	0.59	38.11	22.51	−36.22	147
HONN	0.70	38.10	26.75	−38.71	98
Hybrid-MLP	0.86	38.08	32.80	−59.05	94
Hybrid-RNN	0.81	38.09	30.72	−59.05	93
Hybrid-HONN	0.94	38.07	35.67	−59.05	94

[a]Excluding costs
[b]Annualised

performance measures. The hybrid-RNNs are trained with gradient descent as for the hybrid-MLPs. However, the increase in the number of weights, as mentioned before, makes the training process extremely slow: to derive our results, we needed about ten times the time needed with the hybrid-MLPs. As shown in Table 4.6, the hybrid-RNN has a lower performance compared to the hybrid-MLP model and hybrid-HONN.

We can see that hybrid-HONNs perform significantly better than the hybrid-MLPs and the Hybrid-RNNs and significantly better than the standard neural network architectures despite larger drawdowns. Learning first the linear component of the data-generating process before applying a neural network to learn its non-linear elements definitely appears to add value in this application.

4.5 Trading Costs and Leverage

Up to now, we have presented the trading results of all our models without considering transaction costs. Since some of our models trade quite often, taking transaction costs into account might change the whole picture. Following Dunis et al. [12], we checked for potential improvements to our models through the application of confirmation filters. Confirmation filters are trading strategies devised to filter out those trades with expected returns below the 0.14% transaction cost. These trading strategies examine how the models behave if we introduce a threshold d around zero. They suggest to go long when the forecast is above d and to go short when the forecast is below d. It just so happens that the hybrid ARMA-neural network models perform best without any filter. This is also the case of the MLP and HONN models. Still, the application of confirmation filters to the benchmark models and the RNN model could have led to these models outperforming the hybrid, MLP HONN models. This is not the case in order to conserve space; these results are not shown here but they are available from the authors.

4.5.1 Transaction Costs

According to the Athens Stock Exchange, transaction costs for financial institutions and fund managers dealing a minimum of 143 contracts or 1 million Euros are 10 Euros per contract (round trip). Dividing this transaction cost of the 143 contracts by average size deal (1 million Euros) gives us an average transaction cost for large players of 14 basis points (1 base point = 1/100 of 1%) or 0.14% per position.

From Table 4.7, we can see that, after transaction costs, the hybrid-HONN network outperforms all the other strategies based on the annualised return. The hybrid-MLP strategy performs also well and presents the second best performance in terms of annualised return. It is worth mentioning the good performance of HONN and MLP model. On the other hand, the naïve strategy and the ARMA model seem to be unable to fully exploit the introduction of the modified trading strategy. Furthermore the RNN which also performed well before the introduction of the trading strategy seems also capable of exploiting it. However, the time used to derive these results with the HONN network is half that needed with RNNs and the MLPs.

4.5.2 Leverage to Exploit High Information Ratios

In order to further improve the trading performance of our models we introduce a level of confidence to our forecasts, i.e. a leverage based on the test sub-period. For the naïve model, which presents a negative return, we do not apply leverage. The leverage factors applied are calculated in such a way that each model has a common volatility of 20% on the test data set.[7]

The transaction costs are calculated by taking 0.14% per position into account, while the cost of leverage (interest payments for the additional capital) is calculated at 4% p.a. (i.e. 0.016% per trading day).[8] Our final results are presented in Table 4.8.

[7]Since most of the models have a volatility of about 20%, we have chosen this level as our basis. The leverage factors retained are given in Table 4.8.

[8]The interest costs are calculated by considering a 4% interest rate p.a. divided by 252 trading days. In reality, leverage costs also apply during non-trading days so that we should calculate the interest costs using 360 days per year. But for the sake of simplicity, we use the approximation of 252 trading days to spread the leverage costs of non-trading days equally over the trading days. This approximation prevents us from keeping track of how many non-trading days we hold a position.

Table 4.7 Out-of-sample results with transaction costs

	Information ratio[a]	Annualised volatility[a] (%)	Annualised return[a] (%)	Maximum drawdown[a] (%)	Positions taken[b]	Transaction costs (%)	Annualised return[c] (%)
NAIVE	0.32	36.70	11.42	−49.41	119	16.66	−5.24
MACD	0.46	38.12	17.63	−50.63	38	5.32	12.31
ARMA	0.20	38.13	7.68	−36.50	72	10.08	−2.40
MLP	0.60	38.11	22.99	−36.26	105	14.70	8.29
RNN	0.59	38.11	22.51	−36.22	147	20.58	1.93
HONN	0.70	38.10	26.75	−38.71	98	13.72	13.03
Hybrid-MLP	0.86	38.08	32.80	−59.05	94	13.60	19.20
Hybrid-RNN	0.81	38.09	30.72	−59.05	93	13.02	17.70
Hybrid-HONN	0.94	38.07	35.67	−59.05	94	13.60	22.07

[a]Excluding costs
[b]Annualised
[c]Including costs

Table 4.8 Trading performance—final results

	Information ratio[a]	Annualised volatility[a] (%)	Annualised return[a] (%)	Maximum drawdown[a] (%)	Leverage factor	Positions taken[b]	Transaction and leverage costs (%)	Annualised return[c] (%)
NAIVE	0.32	36.70	11.42	−49.41	–	119	16.66	−5.24
MACD	0.70	40.03	18.51	−53.16	1.050	38	5.60	12.90
ARMA	0.20	38.13	7.68	−36.50	–	72	10.08	−2.40
MLP	0.60	40.28	24.30	−38.32	1.057	105	15.02	9.28
RNN	0.59	40.21	23.75	−38.21	1.055	147	20.88	2.87
HONN	0.70	40.31	28.30	−40.96	1.058	98	14.04	14.26
Hybrid-MLP	0.86	40.14	34.57	−62.24	1.054	94	13.90	20.67
Hybrid-RNN	0.81	40.30	32.50	−62.48	1.058	93	13.34	19.16
Hybrid-HONN	0.94	40.24	37.71	−62.46	1.057	94	13.90	23.21

[a]Excluding costs
[b]Annualised
[c]Including costs

As can be seen from Table 4.8, hybrid-HONNs continue to demonstrate a superior trading performance despite significant drawdowns. The hybrid-MLP strategy also performs well and presents the second higher annualised return. In general, we observe that all models are able to gain extra profits from the leverage as the increased transaction costs seem to counter any benefits. Again it is worth mentioning that the time needed to train the HONN and the hybrid-HONN network was considerably shorter compared with that needed for the MLP, hybrid-MLP, RNN and the Hybrid-RNN networks.

4.6 Concluding Remarks

In this chapter, we applied multilayer perceptron, recurrent, higher order, hybrid-multilayer perceptron, hybrid-recurrent and hybrid-higher order neural networks to a 1-day-ahead forecasting and trading task of the ASE 20 fixing series with only autoregressive terms as inputs. We used a naïve, an MACD and an ARMA model as benchmarks. We developed these different prediction models over the period January 2001–August 2007 and validate their out-of-sample trading efficiency over the following period from September 2007 through December 2008.

The hybrid-HONNs demonstrated the higher trading performance in terms of annualised return and information ratio before transaction costs and elaborate trading strategies are applied. When refined trading strategies are applied and transaction costs are considered again the hybrid-HONNs manage to outperform all other models achieving the highest annualised return. Moreover, the hybrid-MLPs and the hybrid-RNNs models performed remarkably well and seem to have an ability in providing good forecasts when autoregressive series are only used as inputs.

It is also important to note that the hybrid-HONN network which presents the best performance needs less training time than hybrid-RNN and hybrid-MLP network architectures, a much desirable feature in a real-life quantitative investment and trading environment: in the circumstances, our results should go some way towards convincing a growing number of quantitative fund managers to experiment beyond the bounds of traditional statistical and neural network models. In particular, the strategy consisting of modelling in a first stage the linear component of a financial time series and then applying a neural network to learn its non-linear elements appears quite promising.

Appendix

ARMA Model

The output of the ARMA model used in this chapter is presented below.

Dependent variable: RETURNS, Method: Least squares
Sample (adjusted): 81,738, Included observations: 1,731 after adjustments
Convergence achieved after 37 iterations
Backcast: 17

Variable	Coefficient	Std. error	t-statistic	Prob.
C	0.000290	0.000303	0.956602	0.3389
AR(1)	0.375505	0.052705	7.124626	0.0000
AR(3)	−0.244662	0.024991	9.789999	0.0000
AR(7)	−0.678906	0.044902	−15.11958	0.0000
MA(1)	−0.374290	0.053055	7.054702	0.0000
MA(3)	0.269470	0.026409	10.20353	0.0000
MA(7)	0.677169	0.044295	15.28785	0.0000
R-squared	0.026582	Mean dependent var.		0.000288
Adjusted R-squared	0.023194	S.D. dependent var.		0.012549
S.E. of regression	0.012403	Akaike info criterion		−5.937710
Sum squared resid.	0.265213	Schwarz criterion		−5.915645
Log likelihood	5146.088	F-statistic		7.846483
Durbin–Watson stat.	1.857	Prob (F-statistic)		0.000000
Inverted AR roots	$0.89-0.44i$	$0.89+0.44i$	$0.31-0.92i$	$0.31+0.92i$
	$-0.54+0.70i$	$-0.54-0.70i$	-0.93	
Inverted MA roots	$0.88-0.45i$	$0.88+0.45i$	$0.31-0.92i$	$0.31+0.92i$
	$-0.54+0.70i$	$-0.54-0.70i$	-0.94	

Performance Measures

The performance measures are calculated as follows:

Table 4.9 Trading simulation performance measures

Performance measure	Description
Annualised return	$R^A = 252 \times \frac{1}{N}\sum\limits_{t=1}^{N} R_t$
Cumulative return	$R^C = \sum_{t=1}^{N} R_t$
Annualised volatility	$\sigma^A = \sqrt{252}\sqrt{\frac{1}{N-1}\sum\limits_{t=1}^{N}(R_t-\bar{R})^2}$
Information ratio	$IR = R^A/\sigma^A$
Maximum drawdown	Maximum negative value of $\sum R_t$ over the period $MD = \min\limits_{i=1,\cdots,t;t=1,\cdots,N}\left(\sum\limits_{j=i}^{t} R_j\right)$

Statistical Results in the Training and Test Sub-periods

Table 4.10 In-sample statistical performance

	RMSE	MAE	MAPE (%)	THEIL-U
NAIVE	0.0125	0.0125	456.56	0.6781
MACD	0.0131	0.0097	235.17	0.7459
ARMA	0.0124	0.0090	117.82	0.8643
MLP	0.0153	0.0111	371.57	0.6842
RNN	0.0237	0.0119	329.88	0.7174
HONN	0.0141	0.0103	234.72	0.6938
Hybrid-MLP	0.0118	0.0086	108.93	0.7226
Hybrid-RNN	0.0122	0.0082	128.02	0.7760
Hybrid-HONN	0.0118	0.0081	124.91	0.6862

Empirical Results in the Training and Test Sub-periods

Table 4.11 In-sample trading performance

	Information ratio[a]	Annualised volatility[a] (%)	Annualised return[a] (%)	Maximum drawdown[a] (%)	Positions taken[b]
NAIVE	1.55	19.32	29.86	−23.39	114
MACD	1.24	19.49	24.29	−25.42	34
ARMA	1.24	19.83	24.66	−26.70	50
MLP	1.57	19.60	30.72	−27.52	86
RNN	1.53	19.60	30.02	−34.66	81
HONN	1.61	19.59	31.56	−39.70	108
Hybrid-MLP	2.13	19.42	41.35	−37.20	102
Hybrid-RNN	2.01	19.44	39.01	−26.86	79
Hybrid-HONN	2.26	19.40	43.77	−37.20	77

[a]Excluding costs
[b]Annualised

Networks Characteristics

We present below the characteristics of the networks with the best trading performance on the test sub-period for the different architectures.

Table 4.12 Network characteristics for traditional neural networks and hybrid-neural networks

	Learning algorithm	Learning rate	Momentum	Iteration steps	Initialisation of weights	Input nodes	Hidden nodes (1 layer)	Output node
MLP	Gradient descent	0.001	0.003	1,500	N(0,1)	11	7	1
RNN	Gradient descent	0.001	0.003	1,500	N(0,1)	11	6	1
HONNs	Gradient descent	0.001	0.003	1,000	N(0,1)	11	0	1
Hybrid-MLP	Gradient descent	0.001	0.003	1,500	N(0,1)	13	6	1
Hybrid-RNN	Gradient descent	0.001	0.003	1,500	N(0,1)	13	7	1
Hybrid-HONNs	Gradient descent	0.001	0.003	1,000	N(0,1)	13	0	1

References

1. O. Adam, L. Zarader, M. Milgram, Identification and prediction of non-linear models with recurrent neural networks, in *New Trends in Neural Computation*, ed. by J. Mira, J. Cabestany, A. Prieto. Lecture Notes in Computer Science, vol. 686 (Springer, Berlin, 1993), pp. 531–535
2. C. Bishop, Mixture density networks. Technical report NCRG/4288, Neural Computing Research Group, Aston University (1994)
3. G. Box, G. Jenkins, G. Gregory, *Time Series Analysis: Forecasting and Control* (Prentice-Hall, New Jersey, 1994)
4. G. Brown, J. Wyatt, R. Harris, X. Yao, Diversity creation methods: A survey and categorization. Inf. Fusion **6**, 5–20 (2005)
5. R. Clemen, Combining forecasts: A review and annotated bibliography. Int. J. Forecast. **5**, 559–583 (1989)
6. J. Connor, L. Atlas, Recurrent neural networks and time series prediction, in *Proceedings of the International Joint Conference on Neural Networks* (1993), pp. 301–306
7. C. Dunis, Y. Chen, Alternative volatility models for risk management and trading: Application to the EUR/USD and USD/JPY rates. Derivatives Use, Trading and Regulation **11**(2), 126–156 (2005)
8. C. Dunis, X. Huang, Forecasting and trading currency volatility: An application of recurrent neural regression and model combination. J. Forecast. **21**(5), 317–354 (2002)
9. C. Dunis, J. Laws, B. Evans, Modelling and trading the gasoline crack spread: A non-linear story. Derivatives Use, Trading and Regulation **12**, 126–145 (2006)
10. C. Dunis, J. Laws, B. Evans, Trading futures spreads: An application of correlation and threshold filters. Appl. Financ. Econ. **16**, 1–12 (2006)
11. C. Dunis, J. Laws, G. Sermpinis, Modelling and trading the EUR/USD exchange rate at the ECB fixing. Eur. J. Finance **16**(6), 541–560 (2010)
12. C. Dunis, J. Laws, G. Sermpinis, Higher order and recurrent neural architectures for trading the EUR/USD exchange rate. Quant. Finance **11**(4), 615–629 (2011)
13. J.L. Elman, Finding structure in time. Cognit. Sci. **14**, 179–211 (1990)
14. S. Fatima, G. Hussain, Statistical models of KSE100 index using hybrid financial systems. Neurocomputing **7**, 2742–2746 (2008)
15. J. Fulcher, M. Zhang, S. Xu, Application of higher-order neural networks to financial time series, in *Artificial Neural Networks in Finance and Manufacturing*, ed. by J. Kamruzzaman, R. Begg, R. Sarker (Idea Group, Hershey, 2006), pp. 80–108
16. L. Giles, T. Maxwell, Learning, invariance and generalization in higher order neural networks. Appl. Optic. **26**, 4972–4978 (1987)
17. J. Hansen, R. Nelson, Time-series analysis with neural networks and ARIMA-neural network hybrids. J. Exp. Theor. Artif. Intell. **15**(3), 315–330 (2003)
18. H. Hibbert, C. Pedreira, R. Souza, Combining neural networks and arima models for hourly temperature forecast, *in IEEE International Joint Conference on Neural Networks (IJCNN'00)*, vol. 4 (2000), pp. 414–419
19. M. Hibon, T. Evgeniou, To combine or not to combine: Selecting among forecasts and their combinations. Int. J. Forecast. **22**, 15–24 (2005)
20. I. Kaastra, M. Boyd, Designing a neural network for forecasting financial and economic time series. Neurocomputing **10**, 215–236 (1996)
21. K. Kamijo, T. Tanigawa, Stock price pattern recognition: A recurrent neural network approach, in *Proceedings of the International Joint Conference on Neural Networks* (1990), pp. 1215–1221
22. N. Karayiannis, A. Venetsanopoulos, On the training and performance of high-order neural networks. Math. Biosci. **129**, 143–168 (1994)
23. A. Knowles, A. Hussein, W. Deredy, P. Lisboa, C.L. Dunis, Higher-order neural networks with Bayesian confidence measure for prediction of EUR/USD exchange rate, in *Artificial Higher Order Neural networks for Economics and Business*, ed. by M. Zhang (Idea Group, Hershey, 2009), pp. 48–59

24. E. Kosmatopoulos, M. Polycarpou, M. Christodoulou, P. Ioannou, High-order neural network structures for identification of dynamical systems. IEEE Trans. Neural Network **6**, 422–431 (1995)
25. A. Lindemann, C. Dunis, P. Lisboa, Level estimation, classification and probability distribution architectures for trading the EUR/USD exchange rate. Neural Network Comput. Appl. **14**(3), 256–271 (2004)
26. S. Makridakis, Why combining works? Int. J. Forecast. **5**, 601–603 (1989)
27. S. Makridakis, A. Anderson, R. Carbone, R. Fildes, M. Hibdon, R. Lewandowski, J. Newton, E. Parzen, R. Winkler, The accuracy of extrapolation (time series) methods: Results of a forecasting competition. J. Forecast. **1**, 111–153 (1982)
28. P. Newbold, C.W.J. Granger, Experience with forecasting univariate time series and the combination of forecasts (with discussion). J. Stat. **137**, 131–164 (1974)
29. F.C. Palm, A. Zellner, To combine or not to combine? issues of combining forecasts. J. Forecast. **11**, 687–701 (1992)
30. R. Pindyck, D. Rubinfeld, *Econometric Models and Economic Forecasts*, 4th edn. (McGraw-Hill, New York, 1988)
31. D. Psaltis, C. Park, J. Hong, Higher order associative memories and their optical implementations. Neural Network **1**, 149–163 (1988)
32. N. Redding, A. Kowalczyk, T. Downs, Constructive higher-order network algorithm that is polynomial time. Neural Network **6**, 997–1010 (1993)
33. A.F. Shapiro, A hitchhikers guide to the techniques of adaptive nonlinear models. Insur. Math. Econ. **26**, 119–132 (2000)
34. P. Tenti, Forecasting foreign exchange rates using recurrent neural networks. Appl. Artif. Intell. **10**, 567–581 (1996)
35. N. Terui, H. van Dijk, Combined forecasts from linear and nonlinear time series models. Int. J. Forecast. **18**, 421–438 (2002)
36. H. Theil, *Applied Economic Forecasting* (North-Holland, Amsterdam, 1996)
37. P. Tino, C. Schittenkopt, G. Dorner, Financial volatility trading using recurrent neural networks. IEEE Trans. Neural Network **12**(4), 865–874 (2001)
38. F.M. Tseng, H.C. Yu, G.H. Tzeng, Combining neural network model with seasonal time series ARIMA model. Technol. Forecast. Soc. Change **69**, 71–87 (2002)
39. Y.-H. Wang, Nonlinear neural network forecasting model for stock index option price: Hybrid GJR-GARCH approach. Expert Syst. Appl. **36**(1), 564–570 (2009)
40. G.P. Zhang, Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing **50**, 159–175 (2003)
41. R.L. Winkler, Combining forecasts: A philosophical basis and some current issues. Int. J. Forecast. **5**, 605–609 (1989)
42. G.P. Zhang, M. Qi, Neural network forecasting for seasonal and trend time series. Eur. J. Oper. Res. **160**(2), 501–514 (2005)

Chapter 5
Pattern Detection and Analysis in Financial Time Series Using Suffix Arrays

Konstantinos F. Xylogiannopoulos, Panagiotis Karampelas, and Reda Alhajj

Abstract The current chapter focuses on data-mining techniques in exploring time series of financial data and more specifically of foreign exchange currency rates' fluctuations. The data-mining techniques used attempt to analyze time series and extract, if possible, valuable information about pattern periodicity that might be hidden behind huge amount of unformatted and vague information. Such information is of great importance because it might be used to interpret correlations among different events regarding markets or even to forecast future behavior. In the present chapter a new methodology has been introduced to take advantage of suffix arrays in data mining instead of the commonly used data structure suffix trees. Although suffix arrays require high-storage capacity, in the proposed algorithm they can be constructed in linear time $O(n)$ or $O(n\log n)$ using an external database management system which allows better and faster results during analysis process. The proposed methodology is also extended to detect repeated patterns in time series with time complexity of $O(n\log n)$. This along with the capability of external storage creates a critical advantage for an overall efficient data-mining analysis regarding construction of time series data structure and periodicity detection.

5.1 Introduction

The current chapter proposes a method for detecting and analyzing patterns in financial time series using a novel approach. It utilizes the data structure suffix array which is constructed by the time series and is stored in an external database

K.F. Xylogiannopoulos • P. Karampelas (✉)
Hellenic American University, Manchester, NH, USA
e-mail: kostasfx@yahoo.gr; pkarampelas@hauniv.us

R. Alhajj
University of Calgary, Calgary, AB, Canada
e-mail: alhajj@ucalgary.ca

management system. The method searches the suffix array to identify all repeated patterns and finally analyzes the outcome of the search to detect all patterns with specific periodicity. The proposed methodology has been tested by analyzing real financial data for the US Dollar Euro currency exchange rate's values from December 2001 to December 2011 and it has produced interesting results showing that there is scarce periodicity, even in the foreign currency exchange market which is a chaotic system.

A time series is a set of data values representing a variable over a specific time period. The variable can be of any kind such as weather conditions, traffic, banking transactions, stocks or index prices, earthquakes or other geological events, etc. In each distinct fragment of the specific time period examined one value of the variable is assigned. Time period fragmentation can range from nanoseconds in case nuclear phenomena are observed to days for stock markets or even millions of years for geological events. Time series tend to become very important tools in data mining because they can help in discovering periodicities in the events they represent. Periodicities on the other hand can play an important role in decision making, e.g., in profit maximization in financial markets, in cost minimization in operations management in organizations such as banks or supermarkets, in quality of life improvement such as the traffic in large cities, etc.

The first step in the time series processing is to discretize it by assigning a letter of a predefined alphabet to each value or range [3, 16]. Such a process is important before the analysis because:

(a) Ranges of values are more influential than discrete ones since they represent a wide range of values
(b) Ranges can absorb and eliminate noise and errors of the data collection process
(c) Trying to analyze absolute values instead of ranges might be difficult and produce inconsistent results

For example, let $T = e_0, e_1, e_2, \ldots, e_{n-1}$ be a time series of n events, where e_i is the event occurred at time i. Time series T can be discretized by creating an alphabet of m characters to describe each value region. If T is defined from the stock market index daily changes, since there are two significant decimal digits for the change, it means that for a positive change between 0 and 5% there are 501 distinct values. In a time period of 5 years there are approximately 1,260 different values with each value having a probability of almost 0.024 to occur, which if analyzed might not lead to any result. Therefore, the time series can be discretized using a predefined alphabet, denoted by Σ, e.g., by using the following ranges:

- a for change between [0–0.99%]
- b for change between [1.00–1.99%]
- c for change between [2.00–2.99%]
- d for change between [3.00–3.99%]
- e for change between [4.00–5.00%]

Thus, the time series $T = 0.45\%, 2.12\%, 2.44\%, 1.67\%, 3.09\%, 0.87\%$ can be discretized into $T' = accbda$.

Following the discretization process, periodicity detection is applied in two steps by representing data values in memory or other storage means such as a hard disk and analyzing the data. For the representation of data, the most common approach so far is using suffix trees, which is a representation of all suffix strings in a tree data structure [16,22]. A suffix string is a substring of the original string which represents the time series from which a part of the beginning of the string has been removed. For example, let $\tau = \tau_0, \tau_1, \tau_2, \ldots, \tau_n$ be a string with n characters. A suffix string of τ can be $\tau' = \tau_m, \tau_{m+1}, \tau_{m+2}, \ldots, \tau_n$, where $0 \leq m \leq n$. A string of length n can have exactly $n-1$ suffix strings. An alternative approach to achieve the same results is by using, instead of suffix trees, suffix arrays, which is another powerful data structure. A suffix array is a sorted list of all the suffixes of a string [11]. After the creation of the suffix array the analysis of the time series can be done using already developed algorithms for periodicity detection [3, 4, 8, 16].

The rest of the chapter is organized as follows: Section 5.2 reviews the related work. In Sect. 5.3 the problem to be solved is defined. Section 5.4 describes the proposed method. Section 5.5 discusses the findings from applying the proposed method on historical financial data and finally, Sect. 5.6 presents the conclusions and anticipated future work.

5.2 Related Work

The suffix tree of a string is a tree including all the suffixes of the string [17] which is a very powerful data structure [2, 3, 15] heavily used for data-mining analysis because of its flexibility in string processing [6]. Many algorithms have been developed in the past decades to create suffix trees, like Weiner [23] and McCreight [12], with $O(n^2)$ and $O(n \log n)$ complexity, respectively. A suffix tree can also be created in linear time using Ukkonen's algorithm [22] with the assumptions that it can be stored in main memory [2, 6]. Then several other algorithms can be used to traverse suffix trees such as the non-recursive algorithm for binary search tree traversal [1]. In parallel, due to the size of the suffix trees especially in large time series, many techniques have been developed to store them on disks [2, 6]. Nevertheless, performance problems might occur even in the case of the $O(n)$ method of Ukkonen algorithm when traversing the tree especially if the part that is processed is not loaded entirely on memory. In any case, such methods are very useful for processing large time series and their strings [2] such as DNA analysis in bioinformatics and traffic control systems.

Many of the methods for the construction of suffix trees are very time consuming, especially if the time series to be analyzed is very long [2, 21]. Moreover, for very long time series, in which the whole structure has to be stored in a disk instead of memory, significant issues could occur. The constant disk access for writing and reading data can reduce the performance of the algorithms to a great extent [21]. However, the linear time construction and the lesser space consumption compared to other data structures have established suffix trees as the preferable data structure for many string matching analysis tasks [2,15,16,21]. Despite that, many researchers have shown that it is unfeasible to construct a suffix tree that exceeds the available main memory [2, 13, 14]. To overcome this problem some researchers, like Cheung et al. [2], have developed methods to combine on memory and disk data storage and access. Such techniques have a significant improvement of the performance of suffix trees and turn them into a powerful data structure.

Another data structure that has been developed relatively lately is suffix array [9, 17]. It is an array of all the suffixes of a string. The simpler way to construct a suffix array is by using a sorting method such as the merge-sort with complexity of $O(n \log n)$ [9, 11, 17]. Although a suffix array of a string can be constructed in linear time and then it can be lexicographically sorted, Ko and Aluru [10] have developed a method to construct directly the sorted suffix array in $O(n)$ time. The most important disadvantage compared to suffix trees, except construction time, is the storage space required [10, 11]. The elements of a suffix array are $n(n+1)/2$, based on the Gauss proof for the summation of the series $\sum_{k=1}^{n} k$, which is the summation of the elements of all the $n-1$ suffix strings including the original time series string of length n. On the other hand, the advantage of the suffix array is that it does not require memory storing, but instead it can be directly stored on a disk and accessed whenever needed. Although the amount of data to be stored is very large for long strings, since it has space capacity requirement $O(n^2)$, it is more efficient because by taking advantage of the structure of the array it is easier for storing and accessing a massive number of data on a disk instead of the main memory of a computer. Moreover, a database management system can be used for storage, sorting, and accessing and other data operations. Despite the fact that data operations on disk are usually processed slower than in memory, such methods can overcome the usual difficulty to address the problem of the limited memory size by converting it to a more manageable time delay problem due to disk storage as we will present.

For the computation of all the repeating substrings in a time series, methods and algorithms based either on suffix trees or suffix arrays can be used. There are methods that have linear time efficiency such as those described in [6, 20]. These methods can be implemented on suffix trees or suffix arrays and although they might be linear in time consumption and space capacity, the detection of repetitions can be significantly time and space consuming [16,20]. Franek and Smyth have developed an algorithm that can be used on suffix trees and a variation of it can be used on suffix arrays [5] for the computation of all the repeated patterns in a time series.

There are many algorithms that can be used for the analysis of the time series and the detection of any periodicities [3, 15, 16]. One of the earliest was developed by Elfeky et al. [3]. In their work the authors proposed two distinct algorithms for symbol and segment periodicity with complexity $O(n \log n)$ and $O(n^2)$, respectively. Their main difference is that the former does not work properly in the presence of insertion or deletion of noise while the latter does [16]. Many other algorithms have been developed lately using several techniques [4, 8] such as the algorithm of Han et al. [7] for partial periodicity and multiple period data mining in time series. Based on the work of Han et al. [7], Sheng et al. [18, 19] developed an algorithm to detect periodic patterns in a section of a time series [16]. Moreover, Huang and Chang [8] have also developed an algorithm for asynchronous periodic patterns.

Recently, a new approach [15, 16] has been developed which will be used in the current chapter for detecting the periodicity in time series. The specific methods have more efficient results by combining different techniques to report all types of periods even in the presence of noise. In their research, Rasheed et al. [16] have developed periodicity detection algorithms that are very efficient with average complexity $O(n^2)$. Moreover, their algorithms can work in the presence of insertion or deletion of noise and they can also be used for periodicity detection in a subsection of a time series.

5.3 Problem Definition

The data-mining analysis in time series can solve the problem of the detection of all the repeated patterns with a specific periodicity. This problem can be divided into two phases: (a) the detection of all the repeating patterns in the time series and (b) the filtering of the repeated patterns to discover which of them have specific periodicity.

To address these two challenges several techniques have been introduced in the literature. For the identification of all the repeating patterns, suffix trees have been used so far [2, 3, 15, 16] although some methods using suffix arrays have been introduced lately [9, 17]. Both cases require the construction of the data structure first and then the identification of all the existing patterns that have some kind of repetition in time series follows. Subsequent to the retrieval of all the positions of the repeating patterns in the time series, potential periodicities are detected. To detect periodicities, occurrence vectors need to be created first for storing the positions of all repeated patterns. Occurrence vectors are very important because they are required by periodicity detection algorithms [16] in order to analyze the time series and check if these patterns occur with a specific periodicity.

In the current work, suffix arrays will be used instead of suffix trees, for finding all the occurrences of repeated patterns and creating the occurrence vector in the time series. The proposed approach to solve the problem is to first construct the suffix array using a novel methodology and then detect all the repetitions of substrings in it. Subsequently the introduction of an algorithm will follow, which uses the suffix array to produce the occurrence vector of all the patterns.

5.4 Our Approach

The developed methodology is based on the following mathematical definitions and theorems that have been developed and proved for the scope of the chapter. First the definition and calculation of perfect periodicity is introduced. Such a calculation is important because perfect periodicity is needed to introduce confidence limits that will allow us later to detect periodicities that are highly confident and, therefore, valuable for data analysis.

5.4.1 Theorem for the Calculation of Perfect Periodicity of a Subset in a Time Series

Definition 5.1 (Perfect Periodicity). Let a time series $T = \{e_0e_1 \dots e_{n-1}\}$ of $n \in N^*$ elements and length $|T| = n$ and a subset $S = \{e_ie_{i+1} \dots e_{i+k-1}\}$, of the time series T, of $k \in N^*$ elements and length $|S| = k$ that occurs in position i where $0 \leq i \leq i+k-1 \leq n-1$. We define as perfect periodicity PP of the subset S, with period $p \in N$, $p \geq 1$, the maximum number of repetitions that the S can have, with period p, in the time series T.

Theorem 5.1 (Calculation of perfect periodicity). *Let a time series $T = \{e_0e_1 \dots e_{n-1}\}$ of $n \in N^*$ elements and length $|T| = n$ and a subset $S = \{e_ie_{i+1} \dots e_{i+k-1}\}$, of the time series T, of $k \in N^*$ elements and length $|S| = k$. The perfect periodicity PP of the subset S, with period $p \in N$, $p > 1$, can be derived from the formula*

$$PP = \left\lfloor \frac{|T| + (p - |S|)}{p} \right\rfloor.$$

Proof. From the definition of the perfect periodicity we have that PP is the maximum number of repetitions that a subset can have in a time series. Therefore we can write

$$|T| = p \times PP. \tag{5.1}$$

However, the formula is not complete because there are cases in which we can have a residual that we have to add to the multiplication to get the precise number of the length of the time series. Therefore, we can write

$$|T| = p \times PP + R, \tag{5.2}$$

where R is the residual.

From the definition of division between two natural numbers D and d, where D is the dividend and d the divisor, we have

$$D = dq + r. \tag{5.3}$$

What we can notice from comparing (5.2) and (5.3) is that they are analogous. If we change variables D, d, q, and r as follows:

$$D \equiv |T|, \quad d \equiv p, \quad q \equiv PP, \quad r \equiv R$$

then we will have as a result equation (5.2). From this result we can claim that perfect periodicity derives from the algorithm of division since the divisor has exactly the same definition as periodicity, while quotient is the same outcome as perfect periodicity. Indeed with the divisor d we divide the dividend D and get a quotient q, while if we divide $|T|$ with the period p (the divisor) we will get PP (the quotient), plus any remainder.

In the case of perfect periodicity it is important to note that perfect periodicity PP is larger than quotient q by one when the remainder of the division is not 0 (if $r \neq 0$, we do not have perfect division), so, we have to change the equation $PP = q$ to $PP = q + 1 \Rightarrow q = PP - 1$. That happens because while in the division between two natural numbers we get as a result the quotient that gives us how many times D is greater than d (plus a remainder, if any), in periodicity we have to count also the first occurrence of the subset S. Namely, we analyze subsets with a length of at least 1 and therefore the subset itself takes space inside the time series, which is not the case of division between natural numbers. For example, in the division 9/5, we will get as quotient 1 and remainder 4, while in the time series $T_1 = \{a{*}{*}{*}{*}a{*}{*}{*}\}$ with length 9 we can have two occurrences of subset $S = \{a\}$, in positions 0 and 5, with period 5 and a residual of three elements at the end of the time series after the last occurrence of the subset. So we have the same dividend and time series length ($D = |T| = 9$), divisor and period ($d = p = 5$) but we get as a result perfect periodicity $PP = 2$ instead of the quotient $q = 1$ and residual $R = 3$ instead of remainder $r = 4$. The perfect periodicity is greater than quotient by 1 as we expected to be. In the division 10/5 we will get as quotient 2 and remainder 0, while in the time series $T_2 = \{a{*}{*}{*}{*}a{*}{*}{*}{*}\}$ with length 10 we have again two occurrences of subset $S = \{a\}$, in positions 0 and 5, with period 5 and a residual of 4 elements. In this case we have perfect division and that is why perfect periodicity is equal to quotient. If we expand the time series T_2 by adding one more element and create a new time series $T_3 = \{a{*}{*}{*}{*}a{*}{*}{*}{*}a\}$ then we will have three occurrences, at positions 0, 5, and 10, with no residuals, while from the division 11/5 we will get as quotient 2 and remainder 1. Again the perfect periodicity will be greater than quotient by 1 as we have described previously.

Moreover, the above example implies that since the smallest length of the subset we can have is $k = |S| = 1$, the residual R can be $1 \leq R \leq p - 1$, where p is the period. Indeed, although the remainder of the division r is $0 \leq r \leq d - 1$, in the case of perfect periodicity, the residual R can only be $k \leq R \leq p - 1$, $k > 0$ in general or $1 \leq R \leq p - 1$, $k = 1$, which is the smallest value that k can take since $k = |S|$ represents the length of the subset, which cannot be 0. That is because if the residual becomes $R = k - 1$, smaller than k, then the last element of the last occurrence of the subset S will be outside of the time series' boundaries. In that case we will have a reduced perfect periodicity by 1 and the new residual will be $R = p - 1$. Therefore, we can claim that $R \in [k, p - 1]$, $0 < k < p$. For the calculation of perfect periodicity we have to choose the smallest possible R; therefore, we will choose the $R = \min\{k, p - 1\} = k = |S|$ which includes all cases we want to examine. If we choose the greatest possible R, $\max\{k, p - 1\} = p - 1$, we fall into the category of normal division with length of subset $|S| = 1$, which is not the general case since we want to cover all the cases with subsets' length greater than one. Furthermore, if $R = k$ then we have no residual, which is the optimum case we can have for perfect periodicity before it is downgraded to the next smaller integer.

According to the above-mentioned analysis, where PP should be changed to $PP-1$ and R should be replaced with $|S|$, (5.2) can be transformed as follows:

$$|T| = p(PP-1) + |S| \Longrightarrow |T| = pPP - p + |S| \Longrightarrow PP = \frac{|T| + (p - |S|)}{p}. \quad (5.4)$$

Since we care about perfect periodicity, which is a natural number, we can transform (5.4), in order to get the integral part of the outcome, as follows:

$$PP = \left[\frac{|T| + (p-|S|)}{p}\right]$$

which is the biggest natural number smaller than PP, $[PP] \le PP < [PP] + 1$. □

Furthermore, the decimal part that is truncated is R/p and represents the percentage of the remaining elements in the time series following the last occurrence of the substring. It is also showing how much more elements we need in time series to have one more occurrence of the subset since $1 - R/p$ represent the percentage of elements we need to have in a full period. So, the actual number of the elements to complete another period will be $(1 - R/p)p = p - R$.

Example 5.1. Let us take a simple example of calculating perfect periodicity that will demonstrate, why $k \le R \le p-1$ and why we have to choose as R the smallest $R = \min\{k, p-1\} = k = |S|$. Let us assume that we have a time series T_1 as it is presented in Fig. 5.1 with $|T_1| = 20$ and the subset *ab* that starts at position 0 and is repeated with period $p = 5$.

In this case we have perfect periodicity $PP_1 = 4$:

$$PP_1 = \left[\frac{|T_1| + (p-|S|)}{p}\right] = \left[\frac{20 + (5-2)}{5}\right] = [4.6] = 4.$$

Suppose that we truncate the time series to have $|T_2| = 17$. Then we have again perfect periodicity $PP_2 = 4$:

$$PP_2 = \left[\frac{|T_2| + (p-|S|)}{p}\right] = \left[\frac{17 + (5-2)}{5}\right] = [4] = 4.$$

We have to remember that perfect periodicity is not just the quotient but it is by 1 greater since $PP = q+1$. So, if we do the division and express the perfect periodicity as the quotient plus 1 then we will have the following analysis $17/5 = 5 \times 3 + 2 \Rightarrow PP = q + 1 = 3 + 1 = 4$, and, moreover, we have no other values after the subset

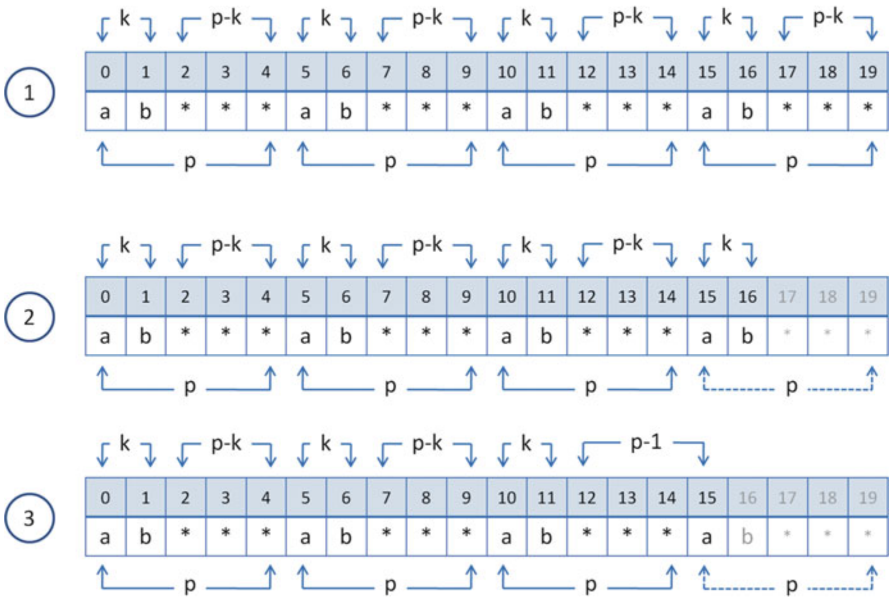


Fig. 5.1 Different time series for the calculation of perfect periodicity

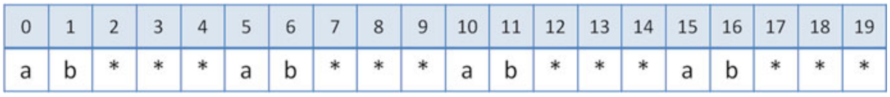
since the remainder of the division $17/5$ is $2 = k$, which is the length of the subset $k = |S|$ and as we have proved it is the smallest value the residual r could take since $k \leq R \leq p - 1, k > 0$.

If we truncate the time series by one more element to have $|T_3| = 16$, then the perfect periodicity will be $PP_3 = 3$

$$PP_3 = \left[\frac{|T_2| + (p - |S|)}{p}\right] = \left[\frac{16 + (5 - 2)}{5}\right] = [3.8] = 3$$

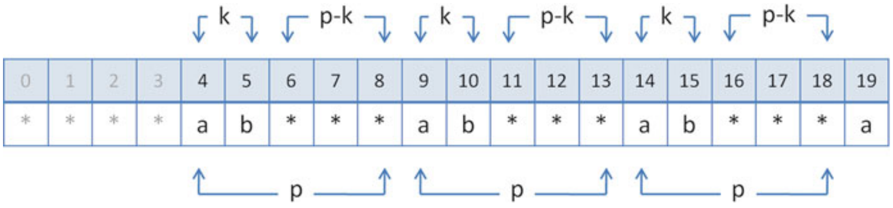
instead of 4 because one character of the subset will be outside of the boundaries of the time series T_3. Now we will have four remaining values (including a at position 15, which is not an occurrence anymore), which is actually the largest value the residual R could take since $k \leq R \leq p - 1, k > 0$ and in this case $p - 1 = 5 - 1 = 4$, while the remainder of the division $16/5$ is $1 < k$. Therefore, we conclude that the optimum case for the perfect periodicity, before we fall into smaller number, is when we have no remaining elements as in the second case in which $R = k = |S|$. □

Lemma 5.1 (Calculation of perfect periodicity starting at a position greater than 0). *Let a time series* $T = \{e_0 e_1 \dots e_{n-1}\}$ *of* $n \in N^*$ *elements and length* $|T| = n$ *and a subset* $S = \{e_i e_{i+1} \dots e_{i+k-1}\}$ *of the time series* T *of* $k \in N^*$ *elements and length* $|S| = k$ *where* $0 \leq i < i + k - 1 \leq n - 1$. *If we want to calculate perfect periodicity for the subset* S *from the position of first occurrence* e_i, *where*



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
a	b	*	*	*	a	b	*	*	*	a	b	*	*	*	a	b	*	*	*

Fig. 5.2 Original time series for calculation of perfect periodicity



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
*	*	*	*	a	b	*	*	*	a	b	*	*	*	a	b	*	*	*	a

Fig. 5.3 The new time series for calculation of perfect periodicity using Lemma 5.1

$i = StartingPosition$, then the perfect periodicity PP of the subset S, with period $p \in N$, $p \geq 1$, can be derived from the formula

$$PP = \left\lfloor \frac{(|T| - i) + (p - |S|)}{p} \right\rfloor .$$

Proof. If we truncate from the time series T, i elements from the beginning then we have a new time series T_1 with length:

$$|T_1| = |T| - StartingPosition = |T| - i$$

which it will give as a result the formula of Lemma 5.1 for the perfect periodicity if we change the new value of $|T_1|$ with its equivalent in the formula of the theorem:

$$PP = \left\lfloor \frac{|T_1| + (p - |S|)}{p} \right\rfloor \Longrightarrow |T_1| = |T| - iPP = \left\lfloor \frac{(|T| - i) + (p - |S|)}{p} \right\rfloor$$

and we get the formula of Lemma 5.1. □

Example 5.2. Let us examine again the first case of Example 5.1 calculating perfect periodicity from a different starting element than e_0 at position 0. We have the time series as presented in Fig. 5.2 in which we have calculated that the perfect periodicity is $PP = 4$. However, assuming that we want to calculate the perfect periodicity starting from position $i = 4$ as it is illustrated in Fig. 5.3 we will have

$$PP = \left\lfloor \frac{(|T| - i) + (p - |S|)}{p} \right\rfloor = \left\lfloor \frac{(20 - 4) + (5 - 2)}{5} \right\rfloor = \lfloor 3.8 \rfloor = 3$$

and we get the expected result because starting from position 4 we have excluded the first occurrence of the subset and we have four free cells, one at the begging and three at the end, which are the most we can have $1+3=4=p-1$. It is like creating a new time series T_1 with $|T_1|=|T|-4=16$, in which the subset starts from position 4 of the first time series T (position 0 of the new time series T_1) with four remaining elements at the end, including a at the position 19 which is not an occurrence any more as it is represented in Fig. 5.3. □

Lemma 5.2 (Calculation of perfect periodicity starting at a position greater than 0 and ending at position less than n − 1). *Let a time series* $T=\{e_0e_1\dots e_{n-1}\}$ *of* $n\in N^*$ *elements and length* $|T|=n$ *and a subset* $S=\{e_ie_{i+1}\dots e_{i+k-1}\}$ *of the time series* T *of* $k\in N^*$ *elements and length* $|S|=k$ *where* $0\le i<i+k-1\le n-1$. *If we want to calculate the perfect periodicity for the subset* S *from the position of first occurrence* e_i*, where* $i=StartingPosition$*, till another position* e_m*, where* $m=EndingPosition$ *and moreover we have that* $0\le i<i+k-1<m\le n-1$*, then the perfect periodicity* PP *of the subset* S*, with period* $p\in N$*,* $p\ge 1$*, can be derived from the formula*

$$PP=\left\lceil\frac{(|T|-i-(|T|-(m+1)))+(p-|S|)}{p}\right\rceil \Longleftrightarrow PP=\left\lceil\frac{(m+1-i)+(p-|S|)}{p}\right\rceil.$$

Proof. If we truncate from the time series T i elements from the beginning and $(|T|-(m+1))$ elements from the end, a new time series T_1 is created with length

$$|T_1|=|T|-StartingPosition-(|T|-(EndingPosition+1))\Longleftrightarrow$$
$$|T_1|=|T|-i-(|T|-(m+1))=m+1-i$$

which it will give as a result the formula of Lemma 5.2 for the perfect periodicity if we change the new value of $|T_1|$ with its equivalent in the formula of the theorem:

$$PP=\left\lceil\frac{|T_1|+(p-|S|)}{p}\right\rceil \Longleftrightarrow |T_1|=m+1-iPP=\left\lceil\frac{(m+1-i)+(p-|S|)}{p}\right\rceil$$

and we get the formula of Lemma 5.2. □

In the second lemma we get the general formula that represents the perfect periodicity of a subset in a time series because if we set $m+1=|T|$ and $i=0$ we fall in the formula of the theorem, while with $m+1=|T|$ and $i\ne 0$ we fall in the formula of the first lemma in which we start from a position greater than 0.

Example 5.3. Let us use again Example 5.2, calculating perfect periodicity from element e_i at position $i=4$ but for a different ending element e_m at the position $m=14$. Then we will have as it is presented in Fig. 5.4 a new time series T_1 starting at position 4 and ending at position 14 of the original time series, in which perfect periodicity will be

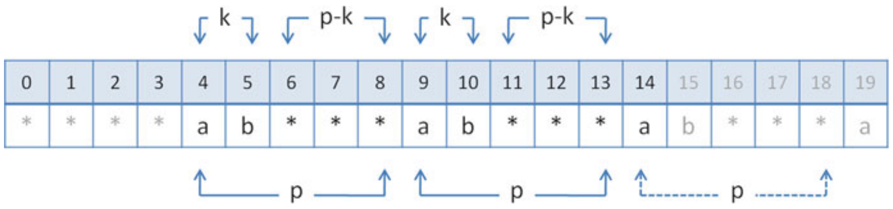


Fig. 5.4 Time series for calculation of perfect periodicity using Lemma 5.2

$$PP = \left[\frac{(|T| - i - ((|T| - 1) - m)) + (p - |S|)}{p} \right] \Longleftrightarrow$$

$$PP = \left[\frac{(m + 1 - i) + (p - |S|)}{p} \right] = \left[\frac{(14 + 1 - 4) + (5 - 2)}{5} \right] = [2.8] = 2$$

and we get the expected result because starting from position 4 and ending at position 14 of the original time series, we can have only two occurrences of the subset with four remaining elements at the end, including a at the position 14 of the original time series or position 10 of the new time series, which is not an occurrence any more. ☐

5.4.2 *Algorithms*

5.4.2.1 Suffix Array Construction

Many different approaches for the construction of the suffix array can be used. The most common is the use of a for-loop structure, which each time removes the first letter of the string. The new substring, which is a suffix string of the initial string of the time series, can be stored either on memory as an array (suffix array) or on the disk. Alternatively, many other algorithms can be used to construct the suffix array in linear time and lexicographically sorted [9, 17]. However, it is important to take into consideration memory storage limitations and constrains. When stored in memory, processing is significantly faster, but, for very long strings, the size of the array might be a drawback because it might exceed the size of the available memory or leave a small amount of free memory for algorithms' operation. In this work in order to calculate occurrence vectors faster and easier, a storage method that uses an external database management system has been selected.

The algorithm Suffix Array Construction (SAC) is used, in which a for-loop calculates each time the substring from the position i till the end of the string of the time series. Then the substring is inserted into database with the respective T-SQL insert command. With the selected algorithm we avoid the construction of a huge array on memory before inserting it into database.

Algorithm 1. Suffix Array Construction
Input: string X of time series
Output: an array of all suffix strings or nothing in case of direct insertion into database

```
1      SAC(string X)
2.1    for i := 0; i<X.length; i++
2.2       subString :=X.Substring(i, X.length - i)
2.3       insert substring into database
2.4    end for
3      end SAC
```

Fig. 5.5 The suffix array and the sorted suffix array of string *abcabbabb*

a

0	a b c a b b a b b
1	b c a b b a b b
2	c a b b a b b
3	a b b a b b
4	b b a b b
5	b a b b
6	a b b
7	b b
8	b

b

6	a b b
3	a b b a b b
0	a b c a b b a b b
8	b
5	b a b b
7	b b
4	b b a b b
1	b c a b b a b b
2	c a b b a b b

Algorithm 1 has only one for-loop and, therefore, it has time complexity $O(n)$. Time complexity cannot be defined with accuracy because the time needed for the insertion of the string into the database might vary on different database management systems and on different software and hardware configurations. In general, the process can be counted as one instruction. In case memory storage is used the time complexity will be $\Theta(4n)$ or generally $O(n)$.

5.4.2.2 Repeated Patterns Detection

Let the time series be represented by the string *abcabbabb*. The length of the time series is $n = 9$. The alphabet of the specific sample is $Alphabet = \{a, b, c\}$ of length $m = 3$. The suffix array of the specific string is represented in Fig. 5.5a.

Irrespectively of the selected type of storage two different types of information for each row of the suffix array table are needed. The first is the position of the suffix string in the time series string and the second is the suffix string. Figure 5.5b represents the lexicographically sorted rows of the table by the suffix string column.

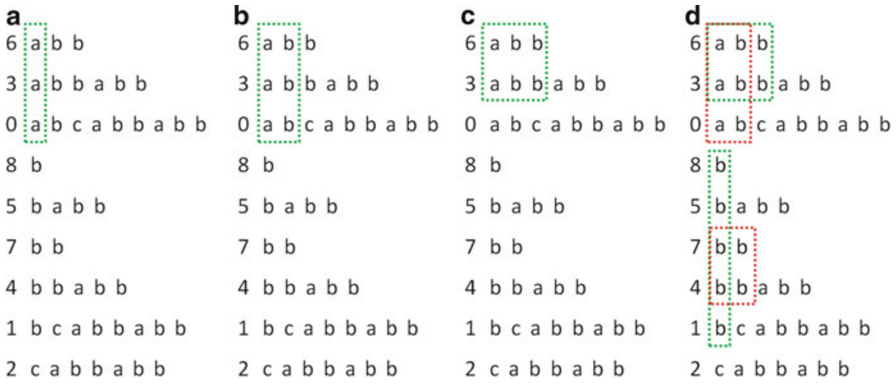


Fig. 5.6 The suffix strings of string *abcabbabb* using a sorted suffix array

In order to calculate the occurrence vectors that will be needed in the periodicity detection algorithms the following process has to be executed:

1. For all the letters of the alphabet count suffix strings that start with the specific letter.
2. If no suffix strings found or only one is found, proceed to the next letter (periodicity cannot be defined with just one occurrence).
3. In case the same number of substrings is found as the total number of the suffix strings, proceed to step 4 and the specific letter is not considered as occurrence because a longer hyper-string will occur.
4. If more than one string and less than the total number of the suffix strings is found, then for the letter used and counted already and for all letters of the alphabet add a letter at the end and construct a new hyper-string. Then do the following checks:

   (a) If none or one suffix string is found that starts with the new hyper-string consider the previous substring as an occurrence and proceed with the next letter of the alphabet.
   (b) If the same number of substrings is found as previously then proceed to step 4. However, the specific substring is not considered as occurrence because a longer hyper-string will occur.
   (c) If more than one and less than the number of occurrences of the previous substring is found, consider the previous substring as a new occurrence and continue the process from step 4.

In the case of the string *abcabbabb* the following process can be used: Starting with first letter of the alphabet, *a*, three substrings can be found that start with a as depicted in Fig. 5.6a. Since more than one and less than the total number of the substrings have been found starting with a, the process should continue to search deeper by adding each letter of the alphabet to a and construct each time a new string.

The first hyper-string is *aa*. Since there is no substring starting with *aa* the process should continue with the next letter and create a new hyper-string, *ab*. Counting the substrings that start with the new string *ab* the result is exactly as many as the previous string (with only one letter, *a*) as illustrated in Fig. 5.6b. In this case, the first string the process started, *a*, is definitely not an important occurrence, because the hyper-string *ab* has occurred exactly the same times in the time series. Since *ab* is longer than *a*, the process uses only the longer *ab*. Continuing to search deeper by adding again each one of the alphabet letters to the new string *ab*, the process starts with the letter *a* and founds no substrings starting with *aba*. It proceeds to the next letter *b*. With the new string *abb* two substrings can be found as presented in Fig. 5.6c. Since the number of the substrings is less than the previous *ab*, definitely *ab* is an occurrence. However, the process should check if *abb* is an occurrence too, or there is a longer string that starts with *abb* is an occurrence. The process continues and finds that there is no string that starts with *abb* and can be counted more than two times. Therefore, the process goes back to step 4 and checks for the string *ac*. Since there are no substrings starting with *ac*, the process has finished with all the substrings starting with *a*. So far the occurrences that are important are *ab* and *abb*. By continuing the process and moving back to the first step and proceeding with the next letter of the alphabet, *b*, the process will find the occurrences *b* and *bb*. For the letter *c* there are no occurrences. So, the whole process has been concluded and produced the findings depicted in Fig. 5.6d.

The whole process can be described by the algorithm Calculate Occurrences' Vectors (COV) "Algorithm 2". The execution of the algorithm should be done by passing an empty string and the length *n* of the time series: COV("", *n*).

In the algorithm there are two external calls: (a) the first one is "how many strings start with newX," which is a T-SQL statement that queries the database and returns the number of the strings that start with the specific substring and (b) the second "find positions of string *X*," which is again a T-SQL statement that gets the positions of the suffix strings in the time series. These positions are the numbers in front of each suffix string. In the specific example with the string *abcabbabb* the occurrences *ab*, *abb*, *b*, *bb* have been found and the equivalent occurrence vectors are: $ab(0,3,6)$, $abb(3,6)$, $b(1,4,5,7,8)$, and $bb(4,7)$.

In the case of memory storage, the appropriate algorithms for sorting and querying the suffix array should be produced to get the respective results, instead of using the two T-SQL statements.

The equivalent suffix tree for the specific sample string will be as it is presented in Fig. 5.7, where \$ is the terminal symbol that is used for each suffix string. The relative occurrence vectors have also the substring positions in parenthesis.

5.4.2.3 Periodicity Detection Algorithms

In order to search and detect if there are any periodicity patterns in the time series, we use the periodicity detection algorithms described thoroughly in the

Algorithm 2. Calculate Occurrences' Vectors
Input: string of pattern we want to check, a counter of string length
Output: an array of all occurrence vectors

```
1        COV(string X, int count)
2        isXcalculated := false
3.1      for each letter l in alphabet
3.2         newX := X + l
3.3         newCount := how many strings start with newX
3.4.1       if newCount = count
3.4.2          COV(newX, newCount)
3.4.3       end if
3.5.1       if (newCount = 1) AND (isXcalculated = false)
                  AND (X NOT null)
3.5.2          find positions of string X
3.5.3          isXcalculated := true
3.5.4       end if
3.6.1       if newCount> 1 AND newCount< count
3.6.2.1        if (isXcalculated = false) AND (X NOTnull)
3.6.2.2           find positions of string X
3.6.2.3           isXcalculated := true
3.6.2.4        end if
3.6.3          COV(newX, newCount)
3.6.4       end if
3.7      end for
4        end COV
```

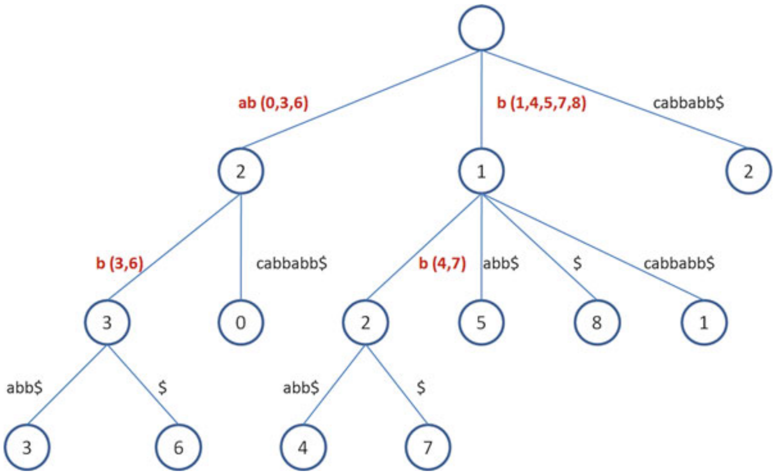


Fig. 5.7 The suffix tree of *abcabbabb* and its occurrences' vectors

research paper [16]. In those algorithms minor modifications have been done using the above-mentioned theorem to calculate confidence and some other minor improvements regarding variable initialization.

Algorithm 3. Periodicity Detection Algorithm
Input: the occurrence vector of time series repeated patterns
Output: positions of periodic patterns

```
1           PDA(string X)
2.1         for each occurrence vector occurVec of size
                   k for pattern X
2.2.1          for j = 0; j < k; j++
2.2.2              p = occurVec[j+1] - occurVec[j]
2.2.3              startPos = occurVec[j]
2.2.4              count = 0
2.2.5.1            for i = j; i< k; i++
2.2.5.2.1              if ((startPos mod p) == (occurVec[i] mod p))
2.2.5.2.2              count++
2.2.5.2.3              end if
2.2.5.3            end for
2.2.6              confidence(p)= count(p) / PP(p,startPos,X)
2.2.7.1            if (confidence(p) >= threshold)
2.2.7.2                add p to the period list
2.2.7.3            end if
2.2.8          end for
2.3         end for
3           end PDA
```

Algorithm 3 searches all the positions for each repeated pattern Algorithm 2 has found to detect periodicity. First it creates the difference vector which is position $i+1$ minus position i of the repeated pattern and calculates the differences (periodicities) p for each pair. Then it checks if the modulo of the division between starting position and p is equal with the modulo of position i and p. If it is the same, it means that the specific position i is a repetition with the specific periodicity p and the algorithm increments the count of the specific periodicity. When finished it calculates the confidence which is the number that periodicity p has been found valid divided by the perfect periodicity.

Perfect periodicity can be calculated using Theorem 5.1. If confidence is equal or larger than the user's specified threshold then the periodicity is added to the list of all valid periodicities. The threshold is the percentage of the lower limit of the confidence. When the process is finished, Algorithm 3 continues to the next occurrence vector.

Algorithm 4 works very similar to Algorithm 3 but allows noise resilience in the time series. The main and very important difference is the introduction of time tolerance value. More specifically, the algorithm searches for periodicities not only in the specific positions that periodicity is expected but also within the limits of a time tolerance. The algorithm namely searches the elements before and after the actual period within a time tolerance value. The new variables that include represent (a) the distance between the current occurrence and the reference starting position

Algorithm 4. Noise Resilient Periodicity Detection Algorithm
Input: the occurrence vector of time series repeated patterns, time tolerance value tt
Output: positions of periodic patterns

```
1            NRPDA(string X, tt)
2.1          for each occurrence vector occurVec
                               of size k for pattern X
2.2.1           for j = 0; j < k; j++
2.2.2              p = occurVec[j+1] - occurVec[j]
2.2.3              startPos = occurVec[j]
2.2.4              currStartPos = StartPos
2.2.5              preOccur = -1
2.2.6              count = 0
2.2.7.1            for i = j; i< k; i++
2.2.7.2               A = occurVec[i] - currStartPos
2.2.7.3               B = Truncate(A/p)
2.2.7.4               C = A - (p*B)
2.2.7.5.1             if ((-tt<= C <= tt) AND
                      (Truncate((preOccur - currStartPos)*p) <> B))
2.2.7.5.2                  currStartPos = occurVec[i]
2.2.7.5.3                  preOccur = occurVec[i]
2.2.7.5.4                  count++
2.2.7.5.5             end if
2.2.7.6            end for
2.2.8              confidence(p)=count(p)/PP(p,startPos,X)
2.2.9.1            if (confidence(p) >= threshold)
2.2.9.2               add p to the period list
2.2.9.3            end if
2.2.10          end for
2.3          end for
3            end NRPDA
```

(variable A), (b) the number of periodic values that must be passed from the current reference starting position to reach the current occurrence (variable B), and (c) the distance between the current occurrence and the expected occurrence (variable C). The preOccur variable holds the value of the current occurrence. If the pattern that is under examination will be found in between the limits of the expected position and the time tolerance (minus or plus) and the current occurrence is not a repetition of an already counted periodic value then the algorithm increments the count that periodicity p has been found [16]. When finished it calculates the confidence which is the number that periodicity p has been found valid divided by the perfect periodicity. Perfect periodicity again can be calculated using Theorem 5.1. If confidence is equal or larger than the user's specified threshold then the periodicity is added to the list of valid periodicities. The threshold is the percentage of the lower limit of the confidence. When the process is finished, Algorithm 4 continues to the next occurrence vector.

Table 5.1 Suffix construction space capacity and time complexity results (real-case scenarios A)

Case	1	2	3	4	5	6	7
Alphabet size (m)	9	9	9	9	9	9	9
String length (n)	100	200	400	800	1,600	3,200	6,400
S.A. construction instructions	698	1,398	2,798	5,598	11,198	22,398	44,798
S.A. space	5,150	20,300	80,600	321,200	1,282,400	5,124,800	20,489,600

5.4.3 Algorithm Analysis

For the data analysis in the current chapter a typical personal computer has been used. The main disadvantage of the suffix array is the storage allocation on memory or on disk. The size of the required storage space is at least $n(n+1)/2$ or $O(n^2)$. For a string of length 100,000 elements the approximate needed disk space is 12 GB in order for the database management system to create the appropriate database file. The need for approximately 2.5 times more space than what the formula $\sum_{k=1}^{n} k = n(n+1)/2$ computes is because of the metadata and other information that the database management system stores in the file. Furthermore, when a field is declared for example, as char(50) in the database management system, even if only one character is stored in the data field, this occupies 50 characters in the file. Database management systems have many techniques to compact database files to their actual size either when the database is constructed or after the process has been completed. Taking this into consideration and the fact that in database management systems the necessary operations for sorting and querying data are available, it is more efficient to store the suffix array in a database than in memory. In addition, the system memory will not be used apart from performing the necessary calculations. Regarding the complexity of the algorithms for the creation of the lexicographically sorted suffix array various results can be produced. The best could be $O(n \log n)$ [9, 17]. However, a simple for-loop method to create the array first can be used and then the database management system can use its internal procedures to lexicographically sort the array. In this case the time complexity will be O(n) for the creation and $O(n \log n)$ for sorting the array, if the database management system uses merge-sort algorithm, which is the most efficient. The overall complexity will be $O(n + n \log n)$ or generally $O(n \log n)$. The relevant calculated results can be found in Table 5.1.

For the Calculate Occurrence Vectors Algorithm, several tests regarding maximum complexity (worst case) and average complexity have been run based on real financial data of Dow Jones Industrial Average Index 30, 1971–2011. Since COV Algorithm uses recursion, it is very difficult to calculate the exact theoretical worst complexity which can be estimated to be $O(10 \times n \times m \times 16 \times \log n)$ or generally $O(n \log n)$. However, so far the experimental findings have shown a time complexity for the worst-case scenario of $O(10 \times n \times m \times 16 \times 2 \times \log m)$ which is almost linear because it can be simplified as $O(n)$, where m is the length of the alphabet and n the length of the string with $m \ll n$. It is very important also to be mentioned that based

Table 5.2 Repeated pattern detection for real-case scenarios of DJIA 30 index (real-case scenarios B)

Case	1	2	3	4	5	6	7
Alphabet size (m)	9	9	9	9	9	9	9
String length (n)	100	200	400	800	1,600	3,200	6,400
Recursions (R)	57	109	204	398	770	1,522	3,154
Occurrences (O)	51	97	182	373	721	1,419	2,924
Instructions (I)	8,063	15,417	28,857	56,390	109,094	215,608	446,717

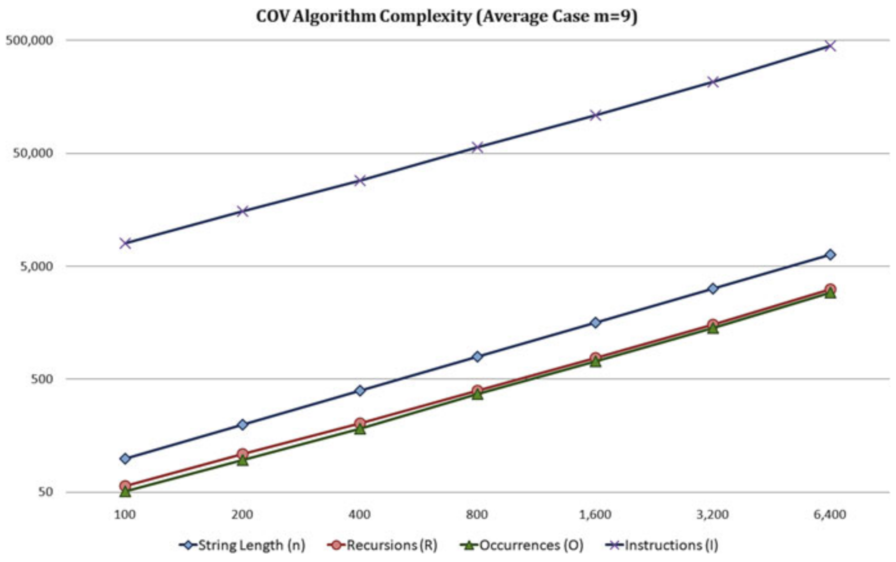


Fig. 5.8 Algorithm's 2 complexity diagram for real data (logarithmic scale)

on real-case scenarios of time series from financial data, the average complexity is $O(\frac{1}{2}mn)$ or generally linear $O(n)$. In both cases the complexity is depending on the alphabet, which is expected since the algorithm uses recursion based on the letters of the alphabet.

Thus, the overall complexity for the creation of the lexicographically sorted suffix array and the execution of the COV algorithm for the detection of the repeated patterns will be of class O$(n + n\log n + n\log n)$ or in general $O(n\log n)$ while for the average case scenario with real data it will be $O(n + n\log n + n)$ or again in general $O(n\log n)$.

Table 5.2 shows some examples from the Dow Jones Industrial Average 30 index for the period 1971–1995 for different time classes from 100 days to almost 25 years data or 6,400 working days. Recursions and of course occurrences are linear analogue to the length of the string, which is important because it shows that the complexity is of type $O(n)$ as illustrated in Fig. 5.8.

Table 5.3 Repeated pattern detection results of mock data for worst-case scenario

Case	1	2	3	4	5
Alphabet size (m)	26	26	26	26	26
String length (n)	208	416	832	1,664	3,328
Recursions (R)	5,356	26,985	70,249	156,777	329,833
Occurrences (O)	136	364	780	1,612	3,276
Instructions (I)	2,100,331	10,579,939	27,541,507	61,432,440	129,226,590

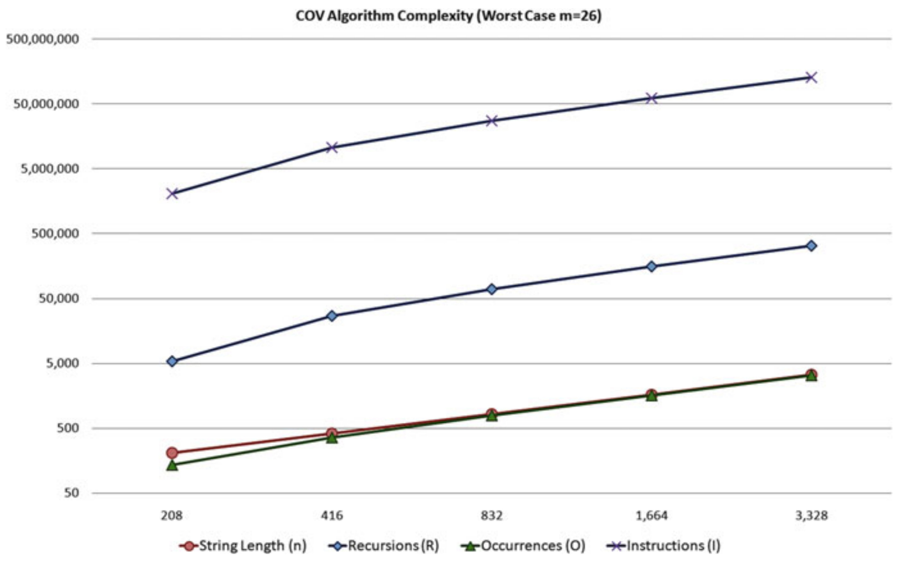


Fig. 5.9 Algorithm's 3 complexity diagram for mock data worst-case scenario (logarithmic scale)

In case of mock data that represent the worst case of a time series for Algorithm 2 the results presented in Table 5.3 can be found. What can be observed from the results is that the instructions and, therefore, the time complexity of Algorithm 2 are almost linear despite the theoretical approach described earlier regarding $O(n \log n)$ complexity. However, the theoretical worst complexity is introduced to avoid underestimation of the time complexity. These results have been illustrated in Fig. 5.9.

The major disadvantage of the described method is the allocation of large storage space on the disk and as a result significant delay in the process because of the slower data access than in memory. However, it can be considered as an advantage since the suffix array is created and stored once in the database and then it can be used in many different ways, e.g., selecting specific time regions to analyze without the need to recalculate each time the respective array. Furthermore, database management systems provide many tools (such as multiprocessing) that can significantly improve the performance of the algorithm. In addition, the alternative method to store the suffix array in memory has a major disadvantage. When computer's memory reaches its limits due to the large amount of data stored and the recursion of the algorithm, then the operating system automatically uses the virtual memory on disk which is significantly slower than the database storing approach.

5.5 Experiments with Financial Data

Suffix arrays and periodicity algorithms have shown some very interesting findings regarding financial data. The most common analysis that can be done is to search if there are specific patterns that occur in currency rate values per day change, by collecting the appropriate time series and then calculating the percentage change of each day from its previous day. Then the percentage daily data change has to be classified by taking into consideration some financial factors. Another type of analysis is to examine also the equivalent weekly data time series instead of daily. In this case, although we lose information, the analysis is more flexible because we can avoid significant noise from the data and its fluctuation.

5.5.1 Data Classification and Classes Construction

Something very important in statistical and financial analysis is the definition of the alphabet. The length of the alphabet (how many discrete ranges exist) and the way it is defined (what are the limits of its range) are very important in order to have credible results.

The most common way to define value regions and therefore the alphabet are, first of all, estimation of the quartiles, deciles, or percentiles. After deciding how many classes are needed or in other words the alphabet used, the calculation of the width of each class follows, in order to discretize the sample and start the analysis.

5.5.2 Financial Aspects

Many statistical tools and methods in stock market analysis, known also as technical analysis, have shown that there are patterns that occur periodically in financial time series and especially major currency rates such as US Dollar and Euro. The most common results fit with the major economic cycles. Each economic cycle can have several growth and recession periods, something that stock markets tend to follow. Moreover, regarding specific major stocks, we can have again results based not only on economic factors but also on sector and market factors.

Periodicity algorithms can detect periodic patterns that might be useful in technical analysis. However, time periods and time ranges are hardly the same, which makes extremely difficult for periodicity detection algorithms to scan and reveal periodicities in standard time series. Furthermore, it has to be mentioned that

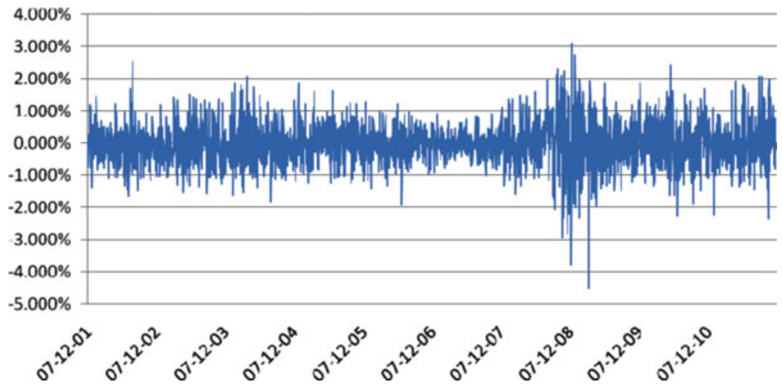


Fig. 5.10 Daily fluctuation of currency rate between US Dollar and Euro

foreign exchange markets are chaotic systems based on many factors that are not necessarily financial (like human behavioral) and it is very difficult to find distinct and clear periodicities like in traffic or other models. More complex models should be defined in order to have credible results. For example, extreme noise might exist in time series because of holiday seasons, different calendar days each Central Bank decides every quarter to report financial facts, figures, interest rate changes and currency policies, even natural disasters like earthquakes, hurricanes or even power failures, which may distort data that otherwise could be periodic. Such an example is the case of the recent earthquake in Japan (summer of 2011) which significantly influenced the US Dollar vs. Japanese Yen currency exchange rate. Moreover, in the case of currency rates, they are directly influenced from many other factors such as inflation and interest rate. Therefore, the development of new approaches, models, and solutions that could eliminate such distortion is needed.

5.5.3 Experimental Results

In the current chapter, data from the US Dollar Euro currency exchange rate will be examined. The specific exchange rate is very important since it represents the two major currencies of the world today. Moreover, because of the financial and state debt crisis of the past 3 years and its further political and economic aspects, the values of the exchange rate have been heavily influenced and present major uncertainty and high volatility in their prices. Currency markets represent transactions of trillion dollars and are very important because they can have significantly high impact on state economic policies, trade, imports, exports, etc. Therefore, it is of great importance to check if there are any patterns that might have periodicity with great confidence.

In the particular research, two different time series will be used for daily changes (2,508 observations) as it is presented in Fig. 5.10 and weekly changes

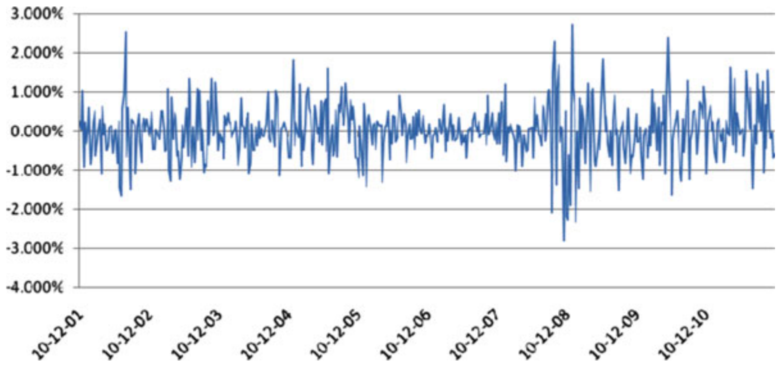


Fig. 5.11 Weekly fluctuation of currency rate between US Dollar and Euro

(518 observations) as depicted in Fig. 5.11. The alphabet used consists of four $\{ABCD\}$ letters for quartiles for each time series. It is also important to mention that in the case of weekly data (instead of more commonly daily), the existence of small patterns of even three or four letters long is very important since they are describing overall longer time periods. A four-letter pattern in weekly data can be considered very important since the underneath time region length is actually 1 month. The data values for the US Dollar Euro currency exchange rate are based on records taken from the Canadian Central Bank's website[1] and time horizon from December 2001 till December 2011.

The first time series of daily data, as it is presented in Fig. 5.10, has an equivalent time series string after applying the alphabet as it is illustrated in Fig. 5.12. The value ranges that each letter represents are: $A = [-4.51, -0.39]$, $B = (-0.39, -0.01]$, $C = (-0.01, 0.34]$, and $D = (0.34, 3.09]$. The second time series of weekly data, represented in Fig. 5.11, has an equivalent time series string after applying the alphabet as it is illustrated in Fig. 5.13. The value ranges that each letter represents are: $A = [-2.81, 0.37]$, $B = (-0.37, 0.02]$, $C = (0.02, 0.36]$, and $D = (0.36, 2.72]$.

In both cases the value of 0.67 has been used as a confidence limit in order to have meaningful results. For example, if a pattern has perfect periodicity 20 and only 2 occurrences, Algorithm 3 will report it as an occurrence; however, for statistical purposes it is not important since it will have confidence 10%. For this reason the confidence has been set to equal 0.67 or greater than 0.67 in order to find more reliable outcomes. It is very important to mention that periodicity detection algorithms have an important effect on analysis. Since they calculate confidence based on the theorem described earlier in the chapter, they tend to lose potentially important results because repetitions used in calculating the confidence start counting from the moment the first pattern occurrence happens till the end. If a pattern starts very early in the time series and also stops very early (in the middle for example) the confidence will be very low. That is not in general bad for the pattern

[1] www.bankofcanada.ca/rates/exchange/10-year-converter.

```
CCBABACCCCDDACAADDCCDACBCDCBCBCDDDABDBACBCAACDCBCADBADDDBCACABDBCBABBCBCDCBDCACBCBDACA
DBBABCCBBACACACABCDABCDABACADCCABADAADACCCBDBCAAAAADAACCCDDAADBCABCCADDBADDADABDADAABA
DBDBBDBBABBCACCDCDCBCCDABBCCCDBACCBDCCACCDBDCBACCBABCAACBCAACDCABBDDCCCCACCABCABCBABCA
CDADAAABDAADBBACBBCABBABABCBDCBACCCDCCACDCBADCBBCBAAABBACCDDDBCBDABBBAABDCCDABADBBABDA
CACDBAACAACAAADCDAABCAACDADCCBADADACABCDCDDDADCABCCDDAADCDCDAACADACCDDBABBDBDBDBDCCDDB
CBCADCAAAABCACDABAABCBCAAABDAABDADDBCBBACBDDCBDDBCDABAABBABCBDBACACABCAABCDABBACBBCABB
ACAADAADDCDDCABADDACDACACAACAACDBACDDBADDCCDDAACADBDABDADADDDDAAAADDAADCDBCACDDBDAADAB
CAADDDCADAACADAACADCBBDAADDBDAADBACCCACBDAACBABCBDACACDDBDBDCBCBBCACBDCACCCADDDCCDBACC
DBABACBDBCCACBCACBACDBCCADDAABCACBACCBCCDAAACCCBCBADACBAABCCBCABBDBCADADBACBABBADDBDDB
BADDCCCDABDADBBCBDCDBBACACBACCACCCADDCACABBADCADDDBDDDBBBDDBBBBADBDAACBBCDCCCBDBBCDBBD
DDCBBCDBBDADDDBCBBBDDDCACADCCDBADBCDBBBCAADCDBDBACCDCABAABDBCCACDCDDCACBABDCAAADCDBDCB
DCDBBDDCDBBBDBAACDBDACDBCCBACADDBADDCCCCDBCBBCBACDACBDBACDACADADBCDCACBCCAABBDCADABDBB
CABCCDCADBDDBCBDCCCCBBCCBDCACABCDBBCAABABCDBDACADABCABDDCBCCABACBBBAAACBCBBCBAABDCDADA
DBCCACCABDCDCDCACBDCADDBBDCABDBADBDCCDCAABDDAABBBBCADBDCCAACDADCCBBCACBCCCDBCBBDBBBBAD
DBCCACCCDDBCCBBDABDBBAACACBDBACABDCCBCBABABBCAABCCCDBBCCDCACBDCBBADDDBCDCBBCBCBCADBDBB
BBDCBBBCCAABCBCCBBCACDBDBBCCABBBACCBADBBCBBCCCADACBBBBBBCCBBDBBDCCBBDBCBBADCBDBDBBBDBC
BBCCDBCCBABBBCBBBCBBABCBCABBBCCBBBCBDBDBBCBACADCDCDDACBBBABCBCCCABABBACCBBAACBBCBABDCC
BDAADBDBACDACAABBBCBCAABCDACBBABBCBCDBDCADCBBCCDDDBDCBBAADACBDBDABACDBDDACADACBABCDBDB
BACABCADABCAABBBBAACBCAABABDDCAABCBDBBACCADBBCACDAADDCBCCDCABDBBACCBAAADACCDDBCDCAACDA
DCABCBADACABCADBDBABABBDBCBDDDBBDCBCDDDDDDCCDBBCADBDBBCDCDDDBDDABCCABABDCCDDCDCDABCDAD
DBDDDBDDBACDBABDBCDCACCDDBAADADDADADACAAAAAADDABCADDBDDACDDDCDADDDADABADDCADCBADDDADDC
ABACCBDDCACACABBAACAADBCBBDDACACDDBDADDBDDABBADDADBABDAAABCCDAABACCDBAAADCDDACADDACABD
ACDABCCCDBDADBCACBACBBDCDDBAACCDCCAADDBACABBDBACDBBCAACBBBABCCACDCCDADABADADAABDBCACBD
DDBDADABCACDBADADCACADCBACBDDDCCDBDADCBDABCCBACBDBACCCDBDDBBCDCDDBBDDDAADDCCADDCBDBDAD
BADBBCABADACDDCBDCAADABDDCCAACADDCCDDACDDAADDDDBADDDDDACAADDACDCDCDCBAACAACBCBDDACCDAA
DADACDABABBDDACACBABADCACDDCDAACCDCBCABDBABADBCBAACACBAACACABCADAACBDAADCDABDADDACCACA
DDDDCDDBABCDCBDDDAAADBDCBABDDDCBCBBAAADDDDCBAACCAADAACBCDAACDDCABDDCBBAABCABCBBAABCDCD
BACCAABBDADCCBABBBACABCBCCDAAACBBABBBCDDDADCDADACDDCCACABCABBDDDCADDACABDBAAAACDDBDDCA
BCDABACCADDADADCABDBBACBDACBCCABDCDDDADDCBBADDABDACABBDDBBAAAADADDADAACDACDDADBCBDBBDC
BCBCDCDABABDAD
```

Fig. 5.12 Time series string for daily data

```
CCDACBCDAACDABBCADBCAACCCABCACAADDDADBACCCACDBACCBCCBCDAACBBBDDCAADAADBDDAAAACABDADCAC
ABDBDACAAADBDDBBDCABBBADCDCCBBCCAAADCCACDAACAACACBCBBCCDBBCCADDBACCCCCBAADACBBDABACDDD
DAADDBCBDBDDADABCABDADDDDCDDCBDCDCAAACADBACDCBCCACCCCABCCDABBDDBBDDDBDCABCBBDADBDCBDCB
BBCABBCBCCBCDACBBDBCBBBCBBBBBACCCBCDCBCBBBBDBDBCDBBCBDBDDADACBCBBBACBCBABAAACCCADBDBBA
DDBBDDBADDADDBCCADAAAADDDABADADCABDDADDAAABADDDBDBABACDBCABCCAAADAAAABDBBCAABCBACADBDB
ADACCDADDDCABCCDBAACACDDABBDDCACDDCDDACDBCAACCBBCABCBBDCBDADCBBCACDDCDAACBDCDBDADADCBB
AA
```

Fig. 5.13 Time series string for weekly data

since it might be considered as an interesting pattern in the time series that might occur again in the future e.g., after a period in time, however, periodicity detection algorithms will discard it if confidence is set to high because of the time gap.

In the first case (Table 5.4) 1,557 distinct patterns have been found with length 1–11. There are some results with zero time tolerance and for patterns with length 3 and 4 characters that occur with great confidence 3 or 4 times in the time series. If we change time tolerance from zero to up to three, more long patterns can be found. That is expected since in this case the time tolerance reduces the noise between the pattern's strings. It is very important to be mentioned though that in the case of daily data, significant findings are not expected due to the high randomness of the data. After all, as being already described and analyzed, foreign exchange markets and financial markets in general are chaotic systems.

Although exchange rates are difficult to be predicted with daily data, several patterns have been found which have some kind of periodicity in which their number is significantly smaller than the total number of occurrences found. It is very important though to analyze the second time series which represents weekly

Table 5.4 Indicative daily data results

Pattern	Starting position	Period	Occurrences	Confidence level	Tolerance
AD	2,387	29	4	0.80	0
AD	2,421	24	3	0.75	0
ADD	1,269	329	3	1.00	0
BD	2,329	53	3	1.00	0
CCA	2,089	142	3	1.00	0
DAD	2,366	52	3	1.00	0
DCCA	293	879	3	1.00	0
D	2,321	9	15	0.71	1
ADD	1,269	329	3	0.75	1
BDA	2,431	24	3	0.75	1
CCA	2,089	142	3	1.00	1
DAD	2,366	52	3	1.00	1
DDA	2,251	110	3	1.00	1
ACBD	1,520	454	3	1.00	1
ADBC	1,903	290	3	1.00	1
CDCC	292	554	3	0.75	1
DCCA	293	879	3	1.00	1
DCDD	2,039	200	3	1.00	1
A	1,826	8	58	0.67	2
B	2,225	18	12	0.75	2
C	1,975	11	33	0.67	2
D	1,313	14	58	0.67	2
DA	2,368	21	6	0.86	2
CDA	2,365	44	3	0.75	2
ACAD	1,534	348	3	1.00	3
BAAA	1,615	259	3	0.75	3
BCBB	1,392	436	3	1.00	3
DDAD	2,417	26	3	0.75	3
DDCC	980	694	3	1.00	3

data, a more compact version of the first case (Table 5.5), as it is mentioned before, weekly data have the ability to normalize the daily data and reduce in a great degree the noise and any abnormal movements because of external factors. Therefore, despite the fact that the COV algorithm has found only 329 distinct patterns in the weekly data, the results from their analysis are much better.

Even without the use of time tolerance, several patterns exist with length up to four characters. It is also important to be mentioned in this case that since each character represents a week, a four-character length pattern represents a month, which is a very long time for exchange rates or stock markets in general. Beside the single character patterns, which are also important since they represent a week, two patterns for three and four characters have been found, and more specifically, ADB and CDA with confidence 0.75, and ADBD and DACB with confidence

Table 5.5 Indicative weekly data results

Pattern	Starting position	Period	Occurrences	Confidence level	Tolerance
A	501	8	3	1.00	0
B	492	11	3	1.00	0
D	469	15	3	0.75	0
DA	466	21	3	1.00	0
DC	374	65	3	1.00	0
DD	238	75	3	0.75	0
CDA	125	103	3	0.75	0
ADBD	249	88	3	0.75	0
DACB	2	157	3	0.75	0
A	441	6	9	0.69	1
B	503	4	3	0.75	1
D	398	10	9	0.75	1
AC	333	63	3	1.00	1
BC	357	42	3	0.75	1
BD	427	29	3	0.75	1
DC	497	7	3	1.00	1
BBC	142	116	3	0.75	1
A	396	5	19	0.76	2
B	115	20	15	0.71	2
C	461	3	14	0.74	2
A	243	6	32	0.70	3
A	324	5	28	0.72	3
B	70	9	34	0.68	3
C	63	14	23	0.70	3
D	36	8	41	0.67	3
D	44	10	33	0.69	3
AB	329	40	4	0.80	3
BC	357	42	3	0.75	3
AAC	447	25	3	1.00	3

0.75, respectively. However, ADB is encapsulated inside the ADBD with different occurrences since the first starts from very early in the time series. Yet, the fact that patterns which represent months in data can be found it is very important and has to be further analyzed in terms of the financial point of view for correlations with other factors, e.g., political.

In the case that time tolerance is used, more interesting results have been identified. More specifically, significant number of occurrences has been found for one-character long patterns. Patterns like A, B, C, and D can be found tenths of times with time tolerance 1, 2, or 3. Furthermore, patterns with two letters have also been found to occur with the time tolerance. Table 5.5 includes some of the findings for the weekly data; however, many more have been found but not included to keep both tables readable.

5.6 Conclusion and Future Work

The current chapter has introduced a new methodology for periodicity detection in time series by using suffix arrays instead of the most commonly used suffix trees. The methodology proposes an algorithm for the construction of the suffix array and another algorithm that searches the sorted suffix array and returns all the repeated patterns. For the calculation of the confidence of the results from the algorithm, perfect periodicity has been introduced. Perfect periodicity calculation is based on the relative theorem that has been proven in this chapter.

Regarding the algorithm complexity, the process of creating a lexicographically sorted suffix array and calculating the repeated patterns (occurrences' vectors) has been computed to be $O(n \log n)$. Especially Algorithm 2, which searches the suffix array to find all the repeated patterns in the time series, has an average complexity of $O(n)$. By using suffix arrays, extremely large time series can be analyzed since they can be stored in a database management system which also has sorting and querying facilities.

The proposed methodology has been applied in US Dollar Euro currency exchange rate which is governed by chaotic models that can be compared more to random walks and therefore there were no great expectations from periodicity detection. The data analysis though has shown that there is some kind of periodicity in several cases which depends on the selected time interval and the alphabet chosen. Moreover, the improvement of the Periodicity Detection Algorithms [16] can lead to more sophisticated and important results, which can trigger off future research for further improvement of both Calculate Occurrences' Vector Algorithm and Periodicity Detection Algorithms.

In future work, new storing approaches will be sought to improve the need for storage space of suffix arrays as it is identified as its main drawback. By achieving less space capacity and in combination of the repeated detection algorithm introduced in this chapter, suffix arrays could be transformed to a powerful tool in time series data mining.

References

1. A. Al-Rawi, A. Lansari, F. Bouslama, A new non-recursive algorithm for binary search tree traversal, in *Proceedings of the 10th IEEE International Conference on Electronics, Circuits and Systems* (IEEE Computer Society, Washington, DC, 2003), pp. 770–773
2. C.-F. Cheung, J.X. Yu, H. Lu, Constructing suffix tree for gigabyte sequences with megabyte memory. IEEE Trans. Knowl. Data Eng. **17**(1), 90–105 (2005)
3. M.G. Elfeky, W.G. Aref, A.K. Elmagarmid, Periodicity detection in time series databases. IEEE Trans. Knowl. Data Eng. **17**(7), 875–887 (2005)
4. M.G. Elfeky, W.G. Aref, A.K. Elmagarmid, WARP: time warping for periodicity detection, in *Proceedings of the 5th IEEE International Conference on Data Mining* (IEEE Computer Society, Washington, DC, 2005), pp. 138–145

5. F. Franek, W.F. Smyth, Y. Tang, Computing all repeats using suffix arrays. J. Automata Languages Combinatorics **8**(4), 579–591 (2003)
6. D. Gusfield, *Algorithms on Strings, Trees, and Sequences* (Cambridge University Press, New York, 1997)
7. J. Han, Y. Yin, G. Dong, Efficient mining of partial periodic patterns in time series database, in *Proceedings of the 15th International Conference on Data Engineering, ICDE '99* (IEEE Computer Society, Washington, DC, 1999), p. 106
8. K.-Y. Huang, C.-H. Chang, SMCA: A general model for mining asynchronous periodic patterns in temporal databases. IEEE Trans. Knowl. Data Eng. **17**(6), 774–785 (2005)
9. J. Kärkkäinen, P. Sanders, S. Burkhardt, Linear work suffix array construction. J. ACM **53**, 918–936 (2006)
10. P. Ko, S. Aluru, Space efficient linear time construction of suffix arrays. J. Discrete Algorithm **3**, 143–156 (2005)
11. U. Manber, G. Myers, Suffix arrays: a new method for on-line string searches, in *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms* (Society for Industrial and Applied Mathematics, Philadelphia, 1990), pp. 319–327
12. E.M. McCreight, A space-economical suffix tree construction algorithm. J. ACM **23**(2), 262–272 (1976)
13. G. Navarro, R. Baeza-Yates, A new indexing method for approximate string matching, in *Proceedings of the 10th Annual Symposium on Combinatorial Pattern Matching*, ed. by G. Goos, J. Hartmanis, J. van Leeuwen, vol. 1645 of Lecture Notes in Computer Science (Springer, Berlin, 1999), pp. 163–185
14. G. Navarro, R. Baeza-Yates, A hybrid indexing method for approximate string matching. J. Discrete Algorithm **1**(1), 205–239 (2000)
15. F. Rasheed, R. Alhajj, Using suffix trees for periodicity detection in time series databases, in *Proceedings of the 4th IEEE International Conference on Intelligent Systems*, vol. 2, pp. 11/8–11/13, Varna, Bulgaria, 2008 Sept. 6–8
16. F. Rasheed, M. Alshalfa, R. Alhajj, Efficient periodicity mining in time series databases using suffix trees. IEEE Trans. Knowl. Data Eng. **22**(20), 1–16 (2010)
17. K.B. Schürmann, J. Stoye, An incomplex algorithm for fast suffix array construction. Software Pract. Ex. **37**(3), 309–329 (2007)
18. C. Sheng, W. Hsu, M.-L. Lee, Efficient mining of dense periodic patterns in time series. Technical report, National University of Singapore, 2005. Technical report TR20/05
19. C. Sheng, W. Hsu, M.-L. Lee, Mining dense periodic patterns in time series data, in *Proceedings of the 22nd International Conference on Data Engineering* (IEEE Computer Society, Washington, DC, 2006), p. 115
20. W.F. Smyth, Computing periodicity in strings – a new approach, in *Proceedings of the 16th Australasian Workshop on Combinatorial Algorithms*, pp. 263–268, Victoria, Australia, 18–21 Sept 2005
21. Y. Tian, S. Tata, R.A. Hankins, J.M. Patel, Practical methods for constructing suffix trees. VLDB J. **14**(3), 281–299 (2005)
22. E. Ukkonen, Online construction of suffix trees. Algorithmica **14**(3), 249–260 (1995)
23. P. Weiner, Linear pattern matching algorithms, in *Proceedings of the 14th Annual Symposium on Switching and Automata Theory* (IEEE Computer Society, Washington, DC, 1973), pp. 1–11

Chapter 6
Genetic Programming for the Induction of Seasonal Forecasts: A Study on Weather Derivatives

Alexandros Agapitos, Michael O'Neill, and Anthony Brabazon

Abstract The last 10 years has seen the introduction and rapid growth of a market in weather derivatives, financial instruments whose payoffs are determined by the outcome of an underlying weather metric. These instruments allow organisations to protect themselves against the commercial risks posed by weather fluctuations and also provide investment opportunities for financial traders. The size of the market for weather derivatives is substantial, with a survey suggesting that the market size exceeded $45.2 Billion in 2005/2006 with most contracts being written on temperature-based metrics. A key problem faced by buyers and sellers of weather derivatives is the determination of an appropriate pricing model (and resulting price) for the financial instrument. A critical input into the pricing model is an accurate forecast of the underlying weather metric. In this study we induce seasonal forecasting temperature models by means of a machine learning algorithm. Genetic Programming (GP) is applied to learn an accurate, localised, long-term forecast of a temperature profile as part of the broader process of determining appropriate pricing model for weather derivatives. Two different approaches for GP-based time series modelling are adopted. The first is based on a simple system identification approach whereby the temporal index of the time-series is used as the sole regressor of the evolved model. The second is based on iterated single-step prediction that resembles autoregressive and moving average models in statistical time-series modelling. The major issue of effective model generalisation is tackled though the use of an ensemble learning technique that allows a family of forecasting models to be evolved using different training sets, so that predictions are formed by averaging the diverse model outputs. Empirical results suggest that GP is able to successfully induce seasonal forecasting models and that search-based autoregressive models

A. Agapitos (✉) • M. O'Neill • A. Brabazon
Financial Mathematics and Computation Research Cluster, Natural Computing Research and Applications Group, Complex and Adaptive Systems Laboratory, University College Dublin, Ireland
e-mail: alexandros.agapitos@ucd.ie; m.oneill@ucd.ie; anthony.brabazon@ucd.ie

compose a more stable unit of evolution in terms of generalisation performance for the three datasets considered. In addition, the use of ensemble learning of 5-model predictors enhanced the generalisation ability of the system as opposed to single-model prediction systems. On a more general note, there is an increasing recognition of the utility of evolutionary methodologies for the modelling of meteorological, climatic and ecological phenomena, and this work also contributes to this literature.

6.1 Introduction

Weather conditions affect the cash flows and profits of businesses in a multitude of ways. For example, energy company sales will be lower if a winter is warmer than usual, leisure industry firms such as ski resorts, theme parks, hotels are affected by weather metrics such as temperature, snowfall or rainfall, construction firms can be affected by rainfall, temperatures and wind levels and agricultural firms can be impacted by weather conditions during the growing or harvesting seasons [30]. Firms in the retail, manufacturing, insurance, transport and brewing sectors will also have weather "exposure." Less obvious weather exposures include the correlation of events such as the occurrence of plant disease with certain weather conditions (i.e. blight in potatoes and in wheat) [48]. Another interesting example of weather risk is provided by the use of "Frost Day" cover by some of the UK town/county councils whereby a payout is obtained by them if a certain number of frost days (when roads would require gritting—with an associated cost) are exceeded. Putting the above into context, it is estimated that in excess of $1 trillion of activity in the US economy is weather-sensitive [21].

A key component of the accurate pricing of a weather derivative are forecasts of the expected value of the underlying weather variable and its associated volatility. The goal of this study is to produce seasonal predictive models by the means of genetic programming (GP) of the stochastic process that describes temperature. On a more general attempt to induce good-generalising seasonal models, an ensemble learning method (bagging) is employed to minimise high-variance models that are often associated with unstable learning algorithms as is the case of GP.

This chapter is organised as follows. Sections 6.2–6.4 provide the background information to the problem domain tackled, as well as to the problem-solving methods employed. Background information is divided into three major parts. These are:

1. Section 6.2 introduces weather derivatives, discusses various methods for pricing these financial instruments and finally motivates the need for seasonal temperature forecasting as part of a more general model for their pricing.
2. Section 6.3 introduces basic prior approaches to the task of seasonal temperature forecasting and distinguishes between a number of possible scenarios in considering the use of weather forecast information for derivative pricing. This section also motivates our choice of time-series index modelling.

3. Section 6.4 reviews the machine learning method of GP and its application to time-series forecasting with an emphasis on weather, climate and ecology forecasting. The major statistical techniques for time-series modelling are also described in this section with the aim of linking these methods with similar frameworks employed by GP-based time-series modelling systems. The ensemble learning method of *bagging* for improving model generalisation is also introduced in this section.

Following the background sections, Sect. 6.5 details our current scope of research. Section 6.6 describes the data utilised, the experimental setup and the evolutionary model development framework adopted. Section 6.7 discusses the empirical findings, and finally Sect. 6.8 draws our conclusions.

6.2 A Brief Introduction to Weather Derivatives

6.2.1 Managing Weather Risk

In response to the existence of weather risk, a series of financial products have been developed in order to help organisations manage these risks. Usually, the organisation that wishes to reduce its weather risk buys "protection" and pays a premium to the seller who then assumes the risk. If the weather event occurs, the risk taker then pays an amount of money to the buyer. The oldest of these financial products are insurance contracts. However, insurance only provides a partial solution to the problem of weather risk as insurance typically concentrates on the provision of cover against damage to physical assets (buildings, machinery) or cash flows which arise from high-risk, low-probability, events such as floods or storm damage. The 1990s saw a convergence of capital and insurance markets and this led to the creation of additional tools for financial weather risk management. One example of this is provided by "catastrophe bonds" whereby a firm issues debt in the form of long-term bonds. The terms of these bonds include a provision that the payment of principal or interest (or both) to bondholders will be reduced in the event of specified natural disasters—thereby transferring part of the risk of these events to the bondholders. This reduction in capital or interest payments would leave the seller with extra cash to offset the losses caused by the weather disaster. As would be expected, the buyers of catastrophe bonds will demand a risk premium in order to compensate them for bearing this weather risk.

The above financial products do not usually provide cover against lower risk, higher probability, events such as the risk of higher than usual rainfall during the summer season, which could negatively impact on the sales and profits of (for example) a theme park. This "gap" in the risk transfer market for weather eventually led to the creation of a market for weather derivatives which allow counterparties to trade weather risks between each other. In essence, weather derivatives are financial products that provide a payout which is related to the occurrence of pre-defined weather events [49]. These derivatives allow commercial organisations to reduce the volatility of future cash flows by hedging against one of the factors which contribute to volatility, namely the weather. Weather derivatives offer several advantages over insurance contracts as unlike insurance cover there is no need to file a claim or prove damages. Weather derivatives also permit a user to create a hedge against a "good" weather event elsewhere. For example, for an agricultural firm, good weather in another location may increase the harvest in that locality, thereby reducing the price that the firm gets for its own produce due to over supply. Weather derivatives also remove the problem of "moral hazard" that can occur under traditional insurance.

In addition to the trading of weather derivatives in order to manage weather risks, substantial trading in weather derivatives markets is driven by the trading of weather risk as an investment product. As weather is not strongly correlated with the systemic risk in general financial markets, weather derivatives represent an asset class which can provide diversification benefits for investors [52]. Weather derivatives also provide short-term traders with speculative investment possibilities as well as opening up cross trading strategies between weather and commodities markets (as both are impacted by weather) [52].

The scale of weather markets can be gleaned from the fifth annual industry survey by the Weather Risk Management Association (WRMA) (a Washington-based trade group founded in 1999) which suggests that the number of contracts transacted globally in the weather market had risen to more than 1,000,000 in the year ending March 2006, with a notional value of $45.2 billion [51].

6.2.2 Development of Market for Weather Derivatives

The earliest weather derivative contracts arose in the USA in 1997 [23]. A number of factors promoted their introduction at this time. Federal deregulation of the power sector created a competitive market for electricity. Before deregulation, utilities had the opportunity to raise prices to customers in the event of weather-related losses. This created a demand for financial products to allow the newly deregulated utilities to hedge against reductions in sales volume, caused by weather (temperature) fluctuations. Most of the early weather derivatives involved utilities and their imprint on the market remains in that the most-heavily traded weather derivatives are still temperature-based (for this reason, this chapter concentrates on temperature-based derivatives). Apart from deregulation of the power sector, the 1997 El Nino brought an unusually mild winter to parts of the USA. Many firms, including heating oil retailers, utilities and clothing manufacturers, saw their revenue dip during what should have been their peak selling season. This enhanced the visibility of weather-related risks. At the same time, the insurance industry faced a cyclical downturn in premium income, and seeking alternative income sources, was prepared to make capital available to hedge weather risks providing liquidity to the fledgling market [23].

The earliest weather derivatives were traded over-the-counter (OTC) as individually negotiated contracts. The absence of market-traded derivatives restricted the liquidity of the OTC market. In September 1999, the Chicago Mercantile Exchange (CME) (www.cme.com) created the first standardised, market-traded, weather derivatives (futures and options) and this led to a notable increase in their use. The CME also acted as a clearing house for all transactions, reducing substantially the counter-party risk faced by market participants. Currently the CME offer weather derivative contracts on a wide variety of underlying weather metrics including temperature, rainfall, snowfall, frost and hurricanes. The most popular contracts are those based on temperature in 24 US cities including Colorado Springs, Las Vegas, Los Angeles, Portland,Sacramento, Salt Lake City, Tucson, Atlanta, Dallas, Houston, Jacksonville, Little Rock, Raleigh, Chicago, Cincinnati, Des Moines, Detroit, Kansas City, Minneapolis, Baltimore, Boston, New York, Philadelphia and Washington, DC. Weather derivatives are also available based on weather events outside the USA.

6.2.3 OTC Weather Derivatives

Weather derivative contracts typically have a number of common attributes [34]:

- A contract period with a specified start and end date
- A defined measurement station (location) at which the weather variable is to be measured
- An index which aggregates the weather variable over the contract period
- A payoff function which converts the index value into a monetary amount at the end of the contract period

Contracts can be sub-divided into three broad categories [11]:

1. OTC weather derivatives
2. Traded weather futures (equivalent to a swap—in essence this is a combined put and call option—each with the same strike price—with each party taking one side)
3. Traded weather options

The earliest weather derivatives were traded OTC as individually negotiated contracts. In OTC contracts, one party usually wishes to hedge a weather exposure in order to reduce cash flow volatility. The payout of the contract may be linked to the value of a weather index on the CME or may be custom-designed. The contract will specify the weather metric chosen, the period (a month, a season) over which it will be measured, where it will be measured (often a major weather station at a large airport), the scale of payoffs depending on the actual value of the weather metric and the cost of the contract. The contract may be a simple “swap” where one party agrees to pay the other if the metric exceeds a predetermined level while the other party agrees to pay if the metric falls below that level. Thus if an energy firm was concerned that a mild winter would reduce demand for power, it could enter into a swap which would provide it with an increasing payout if average temperature over (for example) a month exceeded 66°F. Conversely, to the extent that average temperature fell below this, the energy firm, benefiting from higher power sales, would pay an amount to the counterparty. OTC contracts usually have a fixed maximum payout and therefore are not open ended. As an alternative to swap contracts, contracts may involve call or put options. As an interesting example of an OTC contract, a London restaurant entered into a contract which provided for a payout based on the number of days in a month when the temperature was less than ‘x’ degrees [11]. This was designed to compensate the restaurant for lost outdoor table sales when the weather was inclement.

In the USA, many OTC (and all exchange-traded) contracts are based on the concept of a *degree-day*. A degree-day is the deviation of a day’s average temperature from a reference temperature. Degree days are usually defined as either *Heating Degree Days* (HDDs) or *Cooling Degree Days* (CDDs). The origin of these terms lies in the energy sector which historically (in the USA) used 65 degrees Fahrenheit as a baseline, as this was considered to be the temperature below which heating furnaces would be switched on (a heating day) and above which air-conditioners would be switched on (a cooling day). As a result HDDs and CDDs are defined as

$$HDD = \text{Max (0, } 65^{o}\text{F} - \text{average daily temperature)} \tag{6.1}$$

$$CDD = \text{Max (0, average daily temperature} - 65^{o}\text{F)} \tag{6.2}$$

For example, if the average daily temperature for December 20th is 36°F, then this corresponds to 29 HDDs ($65 - 36 = 29$). The payoff of a weather future is usually linked to the aggregate number of these in a chosen time period (one HDD or CDD is typically worth \$20 per contract). Hence, the payoff to a December contract for HDDs which (for example) trade at 1025 HDDs on 1st December—assuming that there was a total of 1080 HDDs during December—would be \$1,100 (\$20 $\times$ (1080$-$1025)). A comprehensive introduction to weather derivatives is provided by [34].

6.2.4 Pricing a Weather Derivative

A substantial literature exists concerning the pricing of financial derivatives. However, models from this literature cannot be simply applied for pricing of weather derivatives as there are a number of important differences between the two domains. The *underlying* (variable) in a weather derivative (a weather metric) is non-traded and has no intrinsic value in itself (unlike the underlying in a traditional derivative which is typically a traded financial asset such as a share or a bond). It is also notable that changes in weather metrics do not follow a pure random walk as values will typically be quite bounded at specific locations. Standard (arbitrage-free) approaches to derivative pricing (such as the Black–Scholes option pricing model [16]) are inappropriate as there is no easy way to construct a portfolio of financial assets which replicates the payoff to a weather derivative [20].

In general there are four methods used to price weather risk which vary in their sophistication:

1. *Business Pricing*. This approach considers the potential financial impact of particular weather events on the financial performance of a business. This information combined with the degree of risk adverseness of the business (a utility function [25]) can help determine how much a specific business should pay for "weather insurance."
2. *Burn Analysis*. This approach uses historical payout information on the derivative in order to estimate the expected payoff to the derivative in the future. This approach makes no explicit use of forecasts of the underlying weather metric.
3. *Index modelling*. These approaches attempt to build a model of the distribution of the underlying weather metric (e.g., the number of seasonal cumulative HDDs), typically using historical data. A wide variety of forecasting approaches such as time-series models, of differing granularity and accuracy, can be employed. The fair price of the derivative is the expected value based on this, discounted for the time value of money.
4. *Physical models of the weather*. These employ numerical weather prediction models of varying time horizon and granularity. This approach can incorporate the use of Monte Carlo simulation, by generating a large number of probabilistic scenarios (and associated payoffs for the weather derivative) with the fair price of the derivative being based on these, discounted for the time value of money [46].

As with financial asset returns, weather has volatility, and hence, a key component of the accurate pricing of a weather derivative such as an option are forecasts of the underlying weather variable (an estimate of its expected value) and its associated volatility. As can be seen, the latter two methods above explicitly rely on the production of forecasts of the underlying variable using historic and/or current weather forecast information. This chapter focuses on *index modelling*, whereby temperature composes the weather metric of interest. The section that follows contains a brief introduction to the complex task of weather forecasting for the purposes of pricing a weather derivative. Our discussion concentrates on seasonal temperature forecasting.

6.3 Weather Forecasting for Pricing a Weather Derivative

Weather forecasting is a complex process which embeds a host of approaches and associated time horizons. At one end of the continuum we have short-run weather forecasts which typically are based on structural physical models of atmospheric conditions (known as atmospheric general circulation models—AGCMs). These models divide the atmosphere into a series of "boxes" of defined distance in north–south, east–west, and vertical directions. Starting from a set of initial conditions in each box, the evolution of atmospheric conditions is simulated forward in time using these values and the set of equations assumed to explain atmospheric conditions.

As the outputs from these models are sensitive to initial conditions the most common approach is to develop an ensemble forecast (which consists of multiple future weather scenarios, each scenario beginning from slightly different initial conditions). These models usually have good predictive ability up to about 10 days with rapidly reducing predictive ability after that. Forecasts produced by these models are relatively large-scale in nature and hence, to obtain a regional or localised weather forecast, the output from the AGCM must be "downscaled" (this refers to the process of developing a statistical model which attempts to relate large-scale AGCM forecasts to the weather at a specific location). It should be noted that as weather derivatives are usually written for a specific location, course-grained forecasts from AGCMs are not especially useful for weather derivative pricing (at a specific location).

Longer term forecasts having a time horizon beyond 1 month are typically termed *seasonal forecasts* [53]. There are a variety of methods for producing these forecasts ranging from the use of statistical time-series models based on historic data to the use of complex, course-grained, simulation models which incorporate ocean and atmospheric data. Given the range of relevant phenomena it has proven to be a very difficult task to build structural models for accurate long-term seasonal forecasting and non-structural time-series approaches (which bypass atmospheric data and science) can produce long-run forecasts which are at least as good as those produced by structural models once the forecast horizon exceeds a few weeks [46]. Very long-term climate forecasts are also produced by various groups but these are not relevant for the purposes of weather derivative pricing.

In considering the use of weather forecast information for derivative pricing, we can distinguish between a number of possible scenarios. Weather derivatives can often be traded long before the start of the relevant "weather period", which will determine the payoff to the derivative. In this case we can only use seasonal forecasting methods as current short-run weather forecasts have no useful information content in predicting the weather that will arise during the weather period. The second case is that the derivative is due to expire within the next 10 or so days, so the current short-run weather forecast (along with the weather record during the recent past) has substantial information content in pricing the derivative. Obviously the closer the derivative gets to its expiry date, the less important the weather forecast will become, as the payoff to the derivative will have been substantially determined by weather that has already occurred. The final (and most complex) case is where the derivative has several weeks or months left to run in its weather period, hence its value will need to be ascertained using a synthesis of short-run weather forecasts and information from a longer-run seasonal forecast. The process of integrating these sources of information has been the subject of several studies [33].

6.3.1 Prior Approaches to Seasonal Temperature Forecasting

A number of prior studies have examined the prediction of seasonal temperature in the context of pricing weather derivatives. The historical time series of temperatures for a given location exhibits the following characteristics [11]:

1. Seasonality
2. Mean reversion
3. Noise

A simple linear model for capturing the seasonality component is proposed by [11]:

$$T_t^m = A + Bt + C\sin(\omega t + \varphi), \tag{6.3}$$

where T_t^m is the mean temperature at (day) time t, ω represents a phase angle as the maximum and minimum do not necessarily occur on 1st January and 1st July each year, φ represents the period of the seasonal temperature cycle $(2\pi/365)$. Bt permits mean temperature to change each year, allowing for a general warming or cooling trend, and A provides an intercept term. Daily temperatures display marked mean-reversion, and this supports the idea that the process can be modelled using autoregressive methods. These models can capture the key properties of temperature behavior such as seasonality and other variations throughout the year [21]. The variance of temperatures is not constant during the annual cycle, varying between months but remaining fairly constant within each month [11]. In particular, variability of temperature is higher in winter (in the Northern Hemisphere) than in summer. Thus, the noise component is likely to be complex. In [40] they noted that the assumption of the noise component being i.i.d. did not result in reasonable predictions. This could be improved by allowing the distribution of the noise component to vary dynamically. In modeling temperature, attention can be restricted to discrete estimation processes [40]. Although temperature is continually measured, the values used to calculate the temperature metrics of interest (HDDs or CDDs) are discrete, as they both rely on the mean daily temperature.

Seasonal temperature forecasting can be reduced to the task of index modelling as discussed in Sect. 6.2.4. Two major families of *heuristic* and *statistical* time-series modelling methods are described in the next section, with the aim of introducing the general problem-solving framework employed.

6.4 Machine Learning of Time-Series Forecasting Models

Modern machine learning *heuristic* methods for time-series modelling are based on two main natural computing paradigms, those of *Artificial Neural Networks* and *Evolutionary Automatic Programming* (EAP). Both methods rely on a training phase, whereby a set of adaptive parameters or data-structures are being adjusted to provide a model that is able to uncover sufficient structure in training data in order to allow useful predictions. This work makes use of the main thread of EAP that comes under the incarnation of genetic programming.

There are a number of reasons to suppose that the use of GP can prove fruitful in the seasonal modelling of the temperature at a specific location. As noted, the problem of seasonal forecasting is characterised by a lack of a strong theoretical framework, with many plausible, collinear explanatory variables. Rather than attempting to uncover a theoretical cause and effect model of local temperature for each location, this study undertakes a time-series analysis of historical temperature data for the locations of interest. A large number of functional forms, lag periods and recombinations of historic data could be utilised in this process. This gives rise to a high-dimensional combinatorial problem, a domain in which GP has particular potential. The major issue of effective model generalisation is tackled though the use of an ensemble learning technique that allows a family of forecasting models to be evolved using different training sets, so that predictions are formed by averaging the diverse model outputs. This section introduces the GP paradigm and its application to time-series modelling. Special attention is given to the modelling of ecologic and atmospheric data. The dominant statistical time-series modelling methods are also reviewed in an attempt to motivate the forecasting model representations that will be employed as part of the evolutionary learning algorithm in later sections. Finally, ensemble learning and its impact on model generalisation are discussed in the final sub-section.

6.4.1 Genetic Programming

Genetic programming [37,41–43] (GP) is an automatic programming technique that employs an evolutionary algorithm (EA) to search the space of candidate solutions, traditionally represented using expression-tree structures, for the one that optimises some sort of program-performance criterion. The highly expressive representation capabilities of programming languages allow GP to evolve arithmetic expressions that can take the form of regression models. This class of GP application has been termed "Symbolic Regression," and is potentially concerned with the discovery of both the functional form and the optimal coefficients of a regression model. In contrast to other statistical methods for data-driven modelling, GP-based symbolic regression does not presuppose a functional form, i.e. polynomial, exponential, logarithmic, etc., thus the resulting model can be an arbitrary arithmetic expression of regressors [36]. GP-based regression has been successfully applied to a wide range of financial modelling tasks [18].

GP adopts an evolutionary algorithm (EA), which is a class of stochastic search algorithms inspired by principles of *natural genetics* and *survival of the fittest*. The general recipe for solving a problem with an EA is as follows:

1. Define a *representation space* in which candidate solutions (computer programs) can be specified.

2. Design the *fitness criteria* for evaluating the quality of a solution.
3. Specify a *parent selection and replacement* policy.
4. Design a *variation mechanism* for generating offspring programs from a parent or a set of parents.

In GP, programs are usually expressed using hierarchical representations taking the form of syntax-trees, as shown in Fig. 6.1. It is common to evolve programs into a constrained, and often problem-specific, user-defined language. The variables and constants in the program are leaves in the tree (collectively named as terminal set), whilst arithmetic operators are internal nodes (collectively named as function set). In the simplest case of symbolic regression, the function set consists of basic arithmetic operators, while the terminal set consists of random numerical constants and a set of regressor variables. Figure 6.1 illustrates an example expression-tree representing the arithmetic expression $x+(2-y)$.

GP finds out how well a program works by executing it, and then testing its behaviour against a number of test cases, a process reminiscent of the process of black-box testing in conventional software engineering practice. In the case of symbolic regression, the test cases consist of a set of input–output pairs, where a number of input variables represent the regressors and the output variable represents the regressand. GP relies on an error-driven model optimisation procedure, assigning program fitness that is based on some sort of error between the program output value and the actual value of the regressand variable. Those programs that do well (i.e. high fitness individuals) are chosen to be part of a *program variation* procedure and produce offspring programs. The primary program variation procedures that compose the main search operators of the space of computer programs are *crossover* and *mutation*.

The most commonly used form of crossover is *subtree crossover*, depicted in Fig. 6.1. Given two parents, subtree crossover randomly (and independently) selects a cross-over point (a node) in each parent tree. Then, it creates two offspring programs by replacing the subtree rooted at the crossover point in a copy of the first parent with a copy of the subtree rooted at the crossover point in the second parent, and vice versa. Crossover points are not typically selected with uniform probability. This is mainly due to the fact that the majority of the nodes in an expression-tree are leaf-nodes, thus a uniform selection of crossover points leads to crossover operations frequently exchanging only very small amounts of genetic material (i.e. small subtrees). To counteract this tendency, inner-nodes are randomly selected 90% of the time, while leaf-nodes are selected 10% of the time.

The dominant form of mutation in GP is *subtree mutation*, which randomly selects a mutation point in a tree and substitutes the subtree rooted there with a new randomly generated subtree. An example application of the mutation operator is depicted in Fig. 6.1. Another common form of mutation is *point mutation*, which is roughly equivalent to the bit-flip mutation used in genetic algorithms. In point mutation, a random node is selected and the primitive stored there is replaced with a different random primitive of the same rarity taken from the primitive set. When subtree mutation is applied, this involves the modification of exactly one subtree.

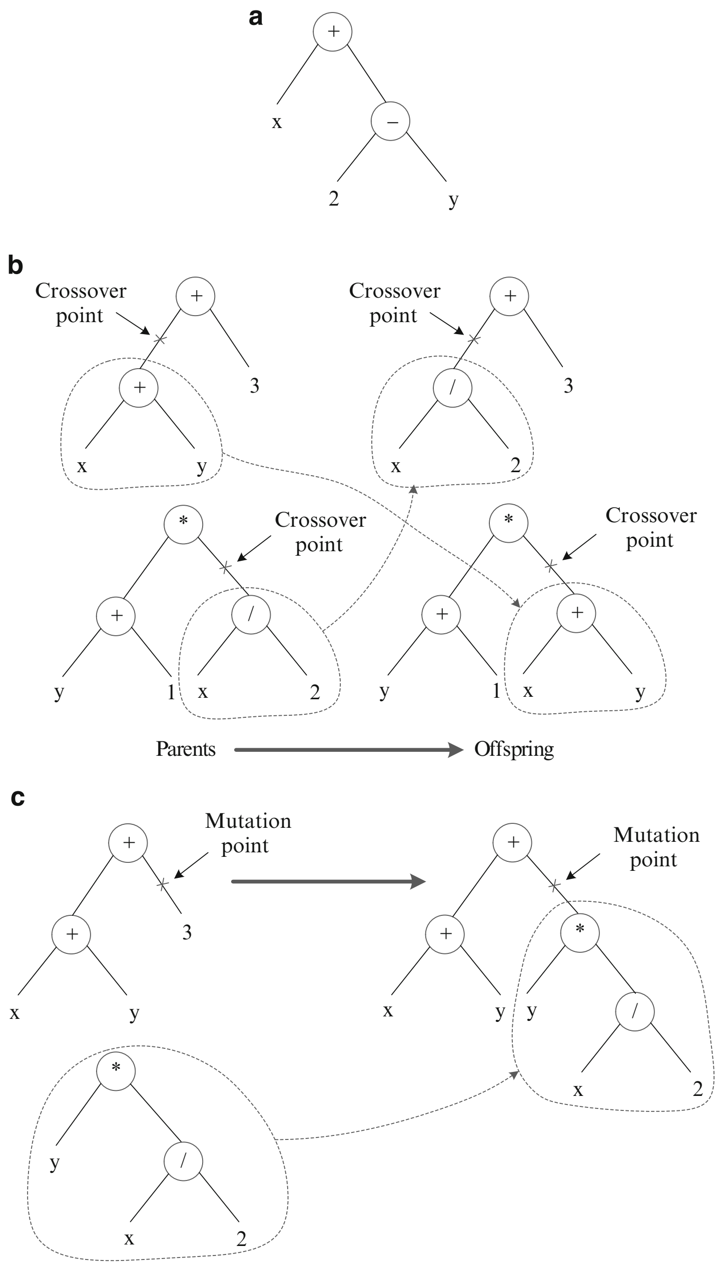


Fig. 6.1 Genetic programming representation and variation operators

Point mutation, on the other hand, is typically applied on a per-node basis. That is, each node is considered in turn and, with a certain probability, it is altered as explained above. This allows multiple nodes to be mutated independently in one application of point mutation.

Like in any EA, the initial population of GP individuals is randomly generated. Two dominant methods are the *full* and *grow* methods, usually combined to form the *ramped half-and-half* expression-tree initialisation method [36]. In both the *full* and *grow* methods, the initial individuals are generated so that they do not exceed a user-specified maximum depth. The depth of a node is the number of edges that need to be traversed to reach the node starting from the tree's root node (the depth of the tree is the depth of its deepest leaf). The *full* method generates full tree structures where all the leaves are at the same depth, whereas the *grow* method allows for the creation of trees of more varied sizes and shapes.

6.4.2 Genetic Programming in Time-Series Modelling

This section describes the approach adopted by GP in time-series forecasting with an emphasis to weather, climate and ecology forecasting. In GP-based time-series prediction [10, 22, 50] the task is to induce a model that consists of the best possible approximation of the stochastic process that could have generated an observed time series. Given *delayed vectors* v, the aim is to induce a model f that maps the vector v to the value x_{t+1}. That is,

$$x_{t+1} = f(v) = f(x_{t-(m-1)\tau}, x_{t-(m-2)\tau}, \dots, x_t), \tag{6.4}$$

where m is embedding dimension and τ is delay time. The embedding specifies on which historical data in the series the current time value depends. These models are known as *single-step predictors* and are used to predict one value x_{t+1} of the time series when all inputs $x_{t-m}, \dots, x_{t-2}, x_{t-1}, x_t$ are given. For long-term forecasts, *iterated single-step prediction models* are employed to forecast further than one step in the future. Each predicted output is fed back as input for the next prediction while all other inputs are shifted back one place. As a result, the input consists partially of predicted values as opposed to observables from the original time series. That is,

$$\begin{aligned}
x'_{t+1} &= f(x_{t-m}, \dots, x_{t-1}, x_t); \quad m < t \\
x'_{t+2} &= f(x_{t-m+1}, \dots, x_t, x'_{t+1}); \quad m < t \\
&\ \ \vdots \\
x'_{t+k} &= f(x_{t-m+k-1}, \dots, x'_{t+k-2}, x'_{t+k-1}); \quad m < t, k \geq,
\end{aligned}$$

where k is the prediction step.

Long-term predictions involve a substantially more challenging task than short-term ones. The fact that each newly predicted value is partially dependent on previously generated predictions creates a reflexive relationship among program outputs, often resulting in inaccuracy propagation and an associated rapid fitness decrease with each additional fitness-case evaluation. Long-term forecasting models are generally sensitive to their initial output values, and inaccuracies of initial predictions are quickly magnified with each subsequent fitness evaluation iteration.

Examining prior literature reveals that evolutionary model induction methodologies have been applied to a number of problems in weather, climate and ecology forecasting. Examples include [24] which used GP to downscale forecasts based on course-grained Atmospheric General Circulation model outputs to estimate local daily extreme (maximum and minimum) temperatures. The results obtained from application of GP to data from the Chute-du-Diable weather station in North Eastern Canada outperformed benchmark results from commonly used statistical downscaling models. GP has also been used for climate prediction problems including rainfall-runoff modelling [54], groundwater level fluctuations [31], short-term temperature prediction [44] and CO_2 emission modelling [12], the combination of ensemble forecasts [14], the forecasting of El Nino [26], evapotranspiration modelling (the process by which water is lost to the atmosphere from the ground surface via evaporation and plant transpiration) [35], modelling the relationship between solar activity and earth temperature [47], stream flow forecasting (forecasting of stream flow rate in a river) [38], modelling of monthly mean maximum temperature [45], modelling of water temperature [13] and wind prediction [29]. Hence we can see that there has been fairly widespread use of GP in this domain, although no previous application to the problem of seasonal forecasting was noted.

6.4.3 Statistical Time-Series Forecasting Methods

Statistical time-series forecasting methods fall into the following five categories; the first three categories can be considered as linear, whereas the last two are non-linear methods:

1. Exponential smoothing methods
2. Regression methods
3. Autoregressive integrated moving average methods (ARIMA)
4. Threshold methods
5. Generalised autoregressive conditionally heteroskedastic methods (GARCH)

In *exponential smoothing*, a forecast is given as a weighted moving average of recent time-series observations. The weights assigned decrease exponentially as the observations get older. In *regression*, a forecast is given as a linear combination of one or more explanatory variables. ARIMA models give a forecast as a linear function of past observations and error values between the time series itself and past observations of explanatory variables. These models are essentially based on a composition of *autoregressive models* (linear prediction formulas that attempt to predict an output of a system based on the previous outputs) and moving average models (linear prediction model based on a *white noise* stationary time series). For a discussion on smoothing, regression and ARIMA methods see [39]. Linear models cannot capture some features that commonly occur in real-world data such as asymmetric cycles and outliers.

Threshold methods [39] assume that extant asymmetric cycles are caused by distinct underlying phases of the time series and that there is a transition period between these phases. Commonly, the individual phases are given a linear functional form, and the transition period is modelled as an exponential or logistic function. GARCH methods [17] are used to deal with time series that display non-constant variance of residuals (error values). In these methods, the variance of error values is modelled as a quadratic function of past variance values and past error values.

Both linear and non-linear methods above, although capable of characterising features such as asymmetric cycles and non-constant variance of residuals, assume that the underlying data-generation process is stationary. For many real-world problems, this assumption is often invalid as shifting environmental conditions may cause the underlying data-generating process to change. In applying the statistical forecasting methods listed above, expert judgement is required to initially select the most appropriate method, and hence select an appropriate model-parameter optimisation technique. In the likely event that the underlying data-generating process is itself evolving, a modelling method must be reevaluated. This is one of the main reasons that forecasting models that can handle dynamic environments are desired.

6.4.4 Ensemble Learning for Model Generalisation

The idea of *supervised ensemble learning* is to induce multiple *base models* and combine their predictions in order to increase *generalisation* performance, i.e. the performance on previously unseen instances. This was originally conceived in the context of *learning algorithm instability*, in which small changes in the training instances can lead to substantially different models with significant fluctuations in accuracy [27]. Ensembles of models approach the phenomenon of overfitting using the statistical concept of *bias-variance tradeoff*, under which the generalisation error of a model is decomposed into the sum of *bias* plus the *variance* [27]. Bias measures the extent to which the learned model is different from the target model, whereas variance measures the extent to which the learned model is sensitive on a particular sample training dataset [15].

There is a trade-off between bias and variance, with very flexible models having low bias and high variance, whereas relatively rigid models having high bias and low variance. To better illustrate the concept of bias and variance, consider that we are constructing a fixed model completely independent of a dataset. In this case, the bias will be high since we are not learning anything from the data; however, the variance will vanish. In the opposite case, where we induce a function that fits the training data perfectly, the bias term disappears, whereas the variance becomes pronounced. Best generalisation is achieved when we have the best balance between the conflicting requirements of small bias and small variance. Ensemble methods are typically based on inducing families of accurate models that are trained on various distributions over the original training dataset. They form an approach to minimise both bias and variance.

A *parallel ensemble* combines independently constructed accurate (low-bias) and diverse (low-variance) base models. In this case, an individual base model is trained on a specific sub-sample of the training instances, and the ultimate requirement is that different base models should make errors of different magnitude when confronted with new instances. Parallel ensembles obtain better generalisation performance than any single one of their components using a variance-reduction technique, and in the majority of cases, they are applied to unstable, high-variance learning algorithms (i.e. decision-tree induction, GP model induction [32]). *Bagging* [19] (bootstrap aggregation) is the earliest parallel ensemble learning method that has been proven very effective for training *unstable* classifiers. The method creates multiple instances of the training dataset by using a bootstrapping technique [28]. Each of these different datasets are used to train a different model. The outputs of the multiple models are hence combined by averaging (in the case of regression) or voting (in the case of classification) to create a single output. Additional methods for enhancing the generalisation of evolved programs have been investigated in [1–9].

6.5 Scope of Research

The goal of this study is to produce predictive models of the stochastic process that describes temperature. More specifically, we are interested in modelling aggregate monthly HDDs using data from three US airport weather stations. Our main objective is to determine whether GP is capable of uncovering sufficient structure in historical data for a series of US locations, to allow useful prediction of the future monthly HDD profile for those locations. The incorporation of the induced models into a complete pricing model for weather derivatives is left for future work. We also restrict attention to the case where the contract period for the derivative has not yet commenced. Hence, we ignore short-run weather forecasts and concentrate on seasonal forecasting.

We investigate two families of program representations for time-series modelling. The first is the standard GP technique, genetic symbolic regression (GSR), applied to the forecasting problem in the same way that it is applied to symbolic regression problems. The task is to approximate a periodic function, where *temperature* (HDDs) is the dependent variable (regressand), and *time* is the sole regressor variable. The second representation allows the induction of iterated single-step predictors that can resemble autoregressive (GP-AR) and autoregressive moving average (GP-ARMA) time-series models that were described in Sect. 6.4.3. In an attempt to provide good-generalising forecasting models, ensembles of predictors are evolved within the general bagging framework for training set resampling and model-output combination. The sections that follow describe the experiment design, discuss the empirical results and draw our conclusions.

6.6 Experiment Design

6.6.1 Model Data

Three US weather stations were selected: (a) Atlanta (ATL); (b) Dallas, Fort Worth (DEN); (c) La Guardia, New York (DFW). All the weather stations were based at major domestic airports and the information collected included date, maximum daily temperature, minimum daily temperature and the associated HDDs and CDDs for the day. This data was preprocessed to create new time series of *monthly* aggregate HDDs and CDDs for each weather station respectively.

There is generally no agreement on the appropriate length of the time series which should be used in attempts to predict future temperatures. Prior studies have used lengths of 20–50 years, and as a compromise this study uses data for each location for the period 01/01/1979–31/12/2002. The monthly HDD data for each location is divided into a *training set* (15 years) that measures the performance during the learning phase and a *test set* (9 years) that quantifies model generalisation.

6.6.2 Forecasting Model Representations and Run Parameters

This study investigates the use of two families of seasonal forecast model representations, where the forecasting horizon is set to 6 months. The first is based on standard GP-based symbolic regression (GSR), where *time* serves as the regressor variable (corresponding to a month of a year), and *monthly HDD* is the regressand variable. Assuming that time t is the start of the forecast, we can obtain a 6-month forecast by executing the program with inputs $\{t+1,\ldots,t+6\}$.

The second representation for evolving seasonal forecasting models is based on the iterated single-step prediction that can emulate autoregressive models, as described in Sect. 6.4.3. This method requires that delayed vectors from the monthly HDD time series are given as input to the model, with each consecutive model output being added at the end of the delayed input vector, while all other inputs are shifted back one place.

Table 6.1 shows the primitive single-type language elements that are being used for forecasting model representation in different experiment configurations. For GSR, the function set contains standard arithmetic operators (protected division) along with e^x, $\log(x)$, $\sqrt{x}$, and finally the trigonometric functions of sine and cosine. The terminal set is composed of the index t representing a month and

Table 6.1 Forecasting model representation languages

Forecasting model	Function set	Terminal set
GSR	add, sub, mul, div, exp,log, sqrt, sin, cos	Index t corresponding to a month 10 rand. constants in $-1.0, \dots, 1.0$ 10 rand. constants in $-10.0, \dots, 10.0$
GP-AR(12)	add, sub, mul, div, exp, log, sqrt	10 rand. constants in $-1.0, \dots, 1.0$ 10 rand. constants in $-10.0, \dots, 10.0$ $HDD_{t-1}, \dots, HDD_{t-12}$
GP-AR(24)	add, sub, mul, div, exp, log, sqrt	10 rand. constants in $-1.0, \dots, 1.0$ 10 rand. constants in $-10.0, \dots, 10.0$ $HDD_{t-1}, \dots, HDD_{t-24}$
GP-AR(36)	add, sub, mul, div, exp, log, sqrt	10 rand. constants in $-1.0, \dots, 1.0$ 10 rand. constants in $-10.0, \dots, 10.0$ $HDD_{t-1}, \dots, HDD_{t-36}$

GP-ARMA(36)	add, sub, mul, div, exp, exp, log, sqrt	10 rand. constants in $-1.0, \dots, 1.0$
		10 rand. constants in $-10.0, \dots, 10.0$
		$HDD_{t-1}, \dots, HDD_{t-36}$
		M($HDD_{t-1},\dots, HDD_{t-6}$), SD($HDD_{t-1},\dots, HDD_{t-6}$)
		M($HDD_{t-1},\dots, HDD_{t-12}$), SD($HDD_{t-1},\dots, HDD_{t-12}$)
		M($HDD_{t-1},\dots, HDD_{t-18}$), SD($HDD_{t-1},\dots, HDD_{t-18}$)
		M($HDD_{t-1},\dots, HDD_{t-24}$), SD($HDD_{t-1},\dots, HDD_{t-24}$)
		M($HDD_{t-1},\dots, HDD_{t-30}$), SD($HDD_{t-1},\dots, HDD_{t-30}$)
		M($HDD_{t-1},\dots, HDD_{t-36}$), SD($HDD_{t-1},\dots, HDD_{t-36}$)

Table 6.2 Learning algorithm parameters

EA	Panmictic, generational, elitist GP with an expression-tree representation
No. of generations	51
Population size	1,000
Tournament size	4
Tree creation	Ramped half-and-half (depths of 2–6)
Max. tree depth	17
Subtree crossover	30%
Subtree mutation	40%
Point mutation	30%
Fitness function	Root mean squared error (RMSE)

random constants within specified ranges. All GP-AR(12), GP-AR(24), GP-AR(36) correspond to standard autoregressive models that are implemented as iterated single-step prediction models. The argument in the parentheses specifies the number of past time-series values that are available as input to the model. The function set in this case is similar to that of GSR excluding the trigonometric functions, whereas

Table 6.3 Comparison of training and testing RMSE obtained by different forecasting configurations, each experiment was run for 50 times

Dataset	Forecasting configuration	Mean training RMSE	Best training RMSE	Mean testing RMSE	Best testing RMSE
ATL	GSR	140.52 (9.55)	68.82	149.53 (8.53)	**72.73**
	GP-AR(12)	92.44 (0.54)	81.78	111.87 (0.41)	103.60
	GP-AR(24)	91.33 (0.68)	83.33	96.15 (0.51)	91.26
	GP-AR(36)	88.96 (0.81)	77.30	90.38 (0.81)	79.44
	GP-ARMA	85.20 (0.86)	75.84	85.71 (0.82)	74.31
DEN	GSR	165.76 (11.46)	103.09	180.46 (11.74)	**95.23**
	GP-AR(12)	133.18 (0.43)	121.38	126.78 (0.25)	117.19
	GP-AR(24)	130.41 (0.73)	111.48	124.36 (0.66)	110.31
	GP-AR(36)	131.13 (1.08)	114.86	111.41 (0.57)	103.73
	GP-ARMA	126.46 (1.29)	106.18	108.90 (0.64)	101.57
DFW	GSR	118.96 (8.02)	66.49	118.69 (7.20)	66.12
	GP-AR(12)	88.75 (0.66)	80.64	90.37 (0.26)	86.57
	GP-AR(24)	96.14 (0.95)	83.55	85.36 (0.42)	78.24
	GP-AR(36)	89.52 (0.69)	81.12	62.11 (0.43)	55.84
	GP-ARMA	87.09 (0.82)	75.41	60.92 (0.52)	**55.10**

Dataset	Forecasting configuration	Ensemble size	Mean training RMSE	Best training RMSE	Mean testing RMSE	Best testing RMSE
ATL	GSR	5	144.90 (4.62)	82.82	150.26 (4.27)	93.29
	GP-AR(12)	5	90.70 (0.38)	84.62	111.40 (0.28)	106.94
	GP-AR(24)	5	85.22 (0.49)	77.32	92.06 (0.29)	88.13
	GP-AR(36)	5	80.01 (0.40)	75.08	80.94 (0.57)	75.65
	GP-ARMA	5	81.60 (0.83)	75.60	80.57 (0.34)	**70.96**
DEN	GSR	5	247.27 (22.70)	121.47	215.87 (7.70)	108.38
	GP-AR(12)	5	131.47 (0.36)	123.37	136.36 (11.13)	120.14
	GP-AR(24)	5	127.64 (0.60)	116.79	122.04 (0.50)	114.35
	GP-AR(36)	5	123.73 (0.86)	110.45	106.42 (0.44)	**92.93**
	GP-ARMA	5	116.86 (0.51)	109.19	109.38 (0.48)	103.49
DFW	GSR	5	165.29 (3.75)	87.93	145.76 (4.05)	75.76
	GP-AR(12)	5	87.11 (0.42)	80.91	89.20 (0.22)	82.71
	GP-AR(24)	5	87.65 (0.49)	80.99	79.21 (0.33)	74.66
	GP-AR(36)	5	86.41 (0.44)	79.74	59.56 (0.33)	**53.07**
	GP-ARMA	5	87.16 (0.60)	77.40	67.20 (0.17)	63.71
ATL	GSR	10	261.62 (18.76)	153.55	190.13 (2.99)	133.39
	GP-AR(12)	10	91.07 (0.30)	85.90	111.71 (0.23)	108.17
	GP-AR(24)	10	85.65 (0.49)	81.25	91.53 (0.21)	88.32
	GP-AR(36)	10	78.82 (0.28)	74.62	79.44 (0.25)	76.43
	GP-ARMA	10	79.95 (0.43)	75.14	80.00 (0.26)	77.67
DEN	GSR	10	295.79 (4.46)	223.11	287.47 (4.73)	203.87
	GP-AR(12)	10	131.20 (0.27)	125.50	125.15 (0.18)	120.60
	GP-AR(24)	10	128.37 (0.41)	122.67	122.59 (0.36)	118.53
	GP-AR(36)	10	122.99 (0.70)	115.29	105.68 (0.31)	101.55
	GP-ARMA	10	116.52 (0.34)	112.35	109.26 (0.37)	104.42

(continued)

Table 6.3 (continued)

Dataset	Forecasting configuration	Ensemble size	Mean training RMSE	Best training RMSE	Mean testing RMSE	Best testing RMSE
DFW	GSR	10	152.20 (5.85)	117.91	144.53 (2.35)	109.15
	GP-AR(12)	10	92.88 (5.21)	83.11	94.55 (0.65)	87.53
	GP-AR(24)	10	87.02 (0.25)	82.93	78.80 (0.18)	76.47
	GP-AR(36)	10	84.98 (0.35)	80.19	58.91 (0.27)	54.32
	GP-ARMA	10	86.97 (0.50)	79.66	66.82 (0.14)	63.70
ATL	GSR	20	245.24 (3.97)	189.02	206.78 (1.79)	178.42
	GP-AR(12)	20	90.76 (0,78)	86.16	110.44 (0.20)	107.24
	GP-AR(24)	20	85.05 (0.24)	82.21	91.21 (0.14)	89.50
	GP-AR(36)	20	78.76 (0.24)	75.95	78.82 (0.13)	77.51
	GP-ARMA	20	79.26 (0.13)	76.18	79.19 (0.16)	76.95
DEN	GSR	20	336.83 (4.43)	286.20	323.56 (3.15)	270.80
	GP-AR(12)	20	131.16 (0.22)	127.63	125.22 (0.14)	123.32
	GP-AR(24)	20	127.53 (0.27)	123.45	121.87 (0.24)	118.17
	GP-AR(36)	20	123.33 (0.52)	115.27	105.91 (0.30)	102.10
	GP-ARMA	20	116.26 (0.40)	111.86	108.52 (0.23)	105.34
DFW	GSR	20	215.47 (2.97)	179.29	189.28 (1.57)	166.87
	GP-AR(12)	20	87.32 (2.09)	82.32	88.90 (0.11)	86.24
	GP-AR(24)	20	85.88 (0.20)	79.72	78.41 (0.12)	76.62
	GP-AR(36)	20	85.40 (0.23)	82.31	59.11 (0.20)	56.43
	GP-ARMA	20	86.37 (0.20)	80.95	67.19 (0.16)	65.19

Standard error for mean is indicated in parentheses. *Bold face* indicates best performance on test data for single base models. *Bold face* combined with *underline* indicates best test performance among all experiment series

the terminal set is augmented with historical monthly HDD values. For the final model configuration, GP-ARMA(36), the function set is identical to the one used in the other autoregressive model configurations; however, the terminal set contains moving averages, denoted by $M(HDD_{t-1},\ldots,HDD_{t-\lambda})$, where λ is the time-lag and HDD_{t-1} and $HDD_{t-\lambda}$ represent the bounds of the moving average period. For every moving average, the associated standard deviation for that period is also given as model input, and is denoted by $SD(HDD_{t-1},\ldots,HDD_{t-\lambda})$. Finally, Table 6.2 presents the parameters of our learning algorithm.

6.6.3 Bagging of GP Time-Series Models

Bagging produces redundant training sets by sampling with replacement from the original training instances. This effectively produces training sets that focus on various distributions over the original learning points. For a number of trials equal to the ensemble size, a training set of equal size to the original training set is sampled with replacement from the original instances. This means that some instances may not appear in it while others appear more than once. An independent GP time-series model is being evolved for every bootstrapped training set, and the outputs of the multiple models are hence combined using a simple averaging procedure in order to predict unseen instances. In this study we are considering ensembles of sizes 5, 10, and 20 independent predictors.

6.7 Results

We performed 50 independent evolutionary runs for each forecasting model configuration presented in Table 6.1. A summary of average and best training and test results obtained using each model configuration is presented in Table 6.3. The first part of the table refers to single-model forecasting, while the second part presents the results obtained by multi-model predictions using different ensemble sizes. The distributions of test-data RMSE obtained by best-of-run models are illustrated in Figs. 6.2–6.4 for ATL, DEN, and DFW datasets, respectively.

For the case of single-model forecasting, the results suggest that the family of autoregressive moving average models performs better on average than those obtained with standard symbolic regression. A statistically significant difference (unpaired t-test, $p < 0.0001$, degrees of freedom $\mathrm{d}f = 98$) was found between the

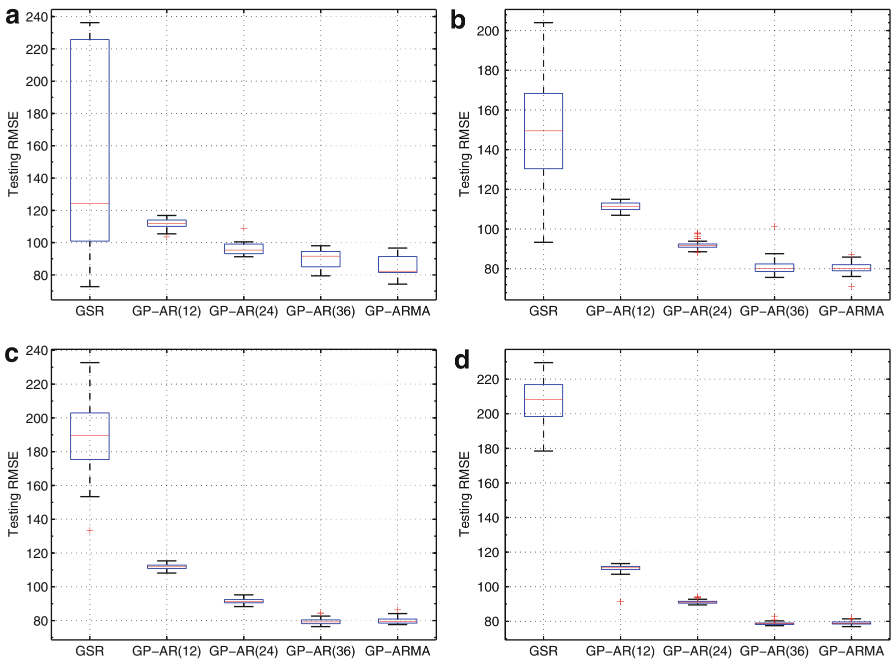


Fig. 6.2 Distribution of best-of-run test RMSE accrued from 50 independent runs for the ATL dataset. (**a**) Single model (**b**) Ensemble size 5 (**c**) Ensemble size 10 (**d**) Ensemble size 20

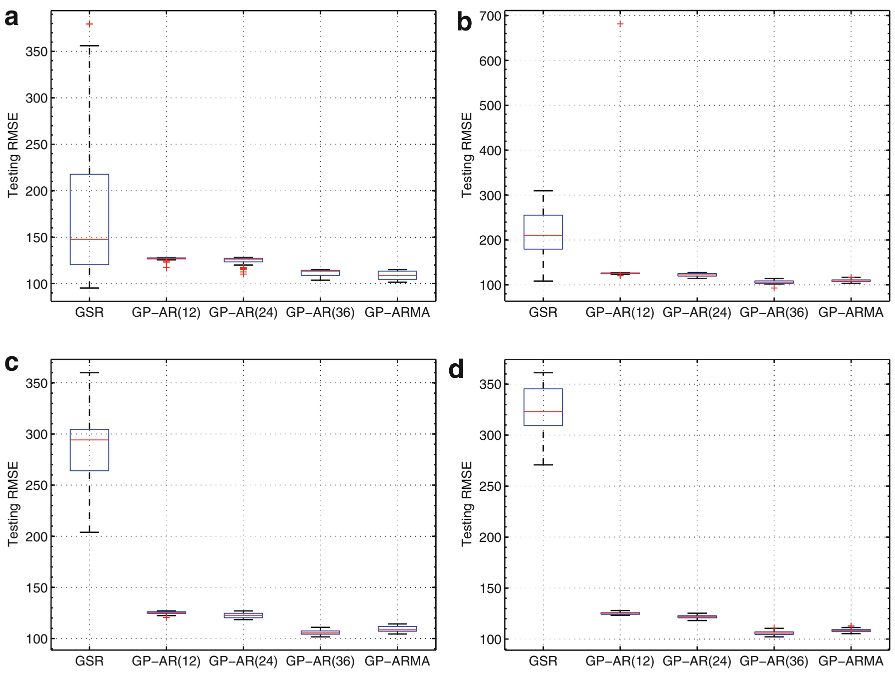


Fig. 6.3 Distribution of best-of-run test RMSE accrued from 50 independent runs for the DEN dataset. (**a**) Single model (**b**) Ensemble size 5 (**c**) Ensemble size 10 (**d**) Ensemble size 20

average test RMSE for GSR and GP-ARMA in all three datasets. Despite the fact that the ARMA representation space offers a more stable unit for evolution than the essentially free-of-domain-knowledge GSR space, best testing RMSE results indicated that GSR models are better performers in ATL and DEN datasets, as opposed to the DFW dataset, where the best-of-50-runs GP-ARMA model appeared superior. Given that in time-series modelling it is often practical to assume a *deterministic* and a *stochastic* part in a series' dynamics, this result can well corroborate on the ability of standard symbolic regression models to effectively capture the deterministic aspect of a time series and successfully forecast future values in the case of time series with a weak stochastic or volatile part. Another interesting observation is that there is a difference in the generalisation performance between GP-AR models of different order, suggesting that the higher the order of the AR process the better its performance on seasonal forecasting. Statistically significant differences (unpaired t-test, $p < 0.0001$, $\mathrm{d}f = 98$) were found in mean test RMSE between GP-AR models of order 12 and those of order 36, in all three datasets. During the learning process, we monitored the test-data performance of the best-of-generation individual, and we adopted a model selection strategy whereby the best-generalising individual from all generations is designated as the outcome of the run. Figure 6.5a–c illustrates the distributions of the generation number where

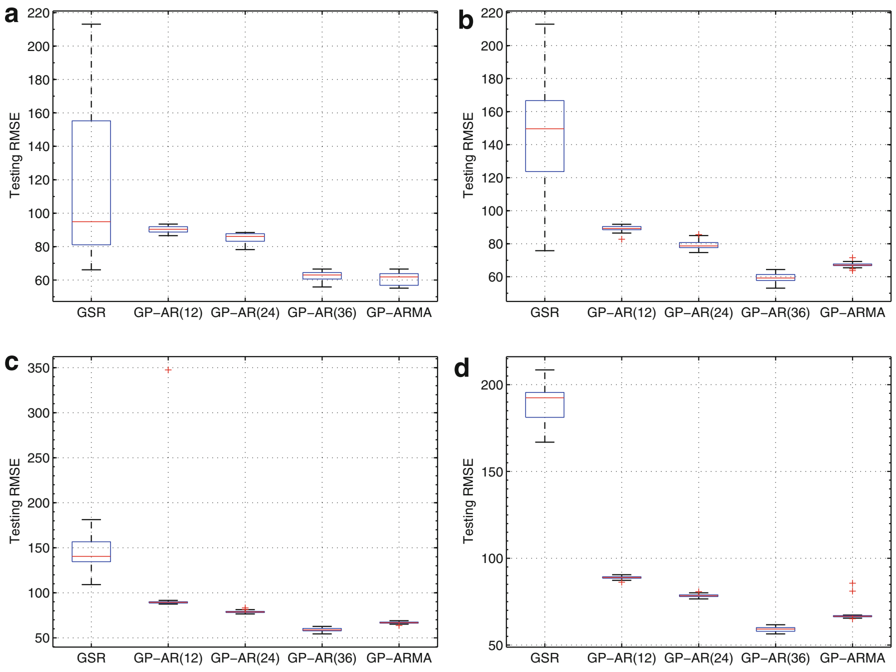


Fig. 6.4 Distribution of best-of-run test RMSE accrued from 50 independent runs for the DFW dataset. (**a**) Single model (**b**) Ensemble size 5 (**c**) Ensemble size 10 (**d**) Ensemble size 20

model selection was performed, for the three datasets. It can be seen that GSR models are less prone to overfitting, then follows GP-ARMA, and finally it can be noted that GP-AR models of high order are the most sensitive to overfitting the training data. Interestingly this fact is observed across all three datasets. In addition to this observation, Fig. 6.6 illustrates the RMSE curves during training. It can be seen that under the GSR model configuration, there is a slower rate of training-error minimisation, with initial models being poorer performers compared to the respective ones under the GP-AR and GP-ARMA model configurations. Eventually, however, we observe that all model configurations reach the same training-error rates. This observation makes the GP-AR and GP-ARMA model configurations much more efficient in terms of search effort required to find the best-of-run generalising models, however, rendering any additional training prone to overfitting.

Looking at the results of Table 6.3 obtained with multi-model predictors, we observe that ensembles of size 5 generalised the best in all datasets, improving the results upon single-model predictors. Interestingly, the best-generalising ensemble GP-AR and GP-ARMA models outperformed their GSR counterparts in all datasets. Statistically significant differences (unpaired t-test, $p < 0.0001$, $\mathrm{d}f = 98$) were found between the mean test RMSE of ensembles of size 5 of autoregressive models and standard symbolic regression models. This is mainly attributed to the unstable performance of GSR models indicated by the high variance in test RMSE in

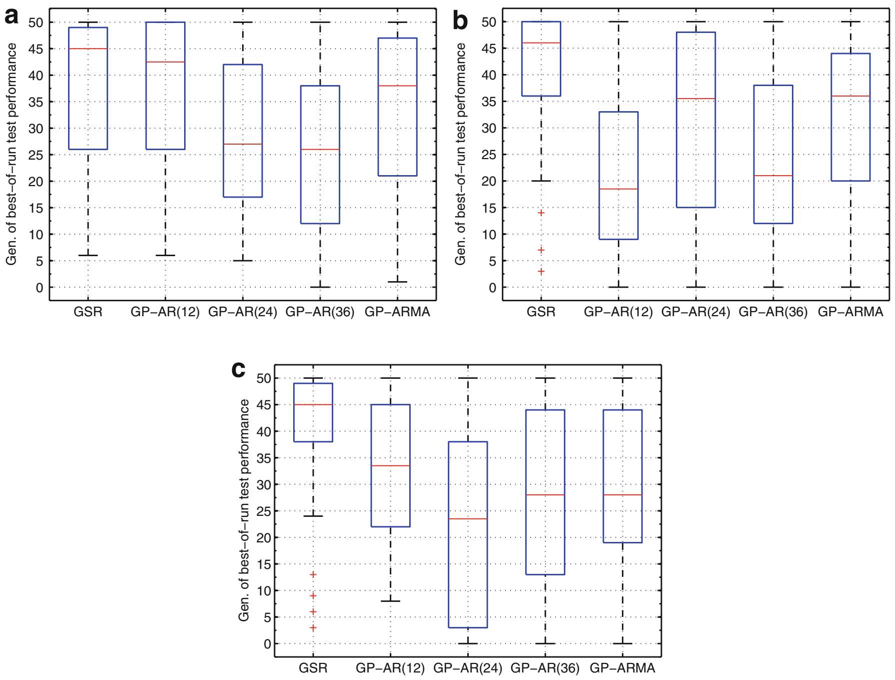


Fig. 6.5 (**a**), (**b**), (**c**) show the distribution of generation number where each best-of-run individual on test data was discovered for the cases of ATL, DEN, and DFW, respectively. Cases for single-model predictions

different evolutionary runs (Figs. 6.2a, 6.3a, 6.4a) and the fact the bagging generates models from resampling the training data and learning models using each subsample separately. An additional interesting observation is that the use of greater ensemble size has an effect in reducing the RMSE variance in the case of GSR; however, increasing the ensemble size shows no pronounced effect in the variance of autoregressive models. Overall, it is noted that increasing the ensemble size beyond 5 models results in worsening the generalisation performance. This observation is consistent across all datasets.

Finally, Figs. 6.7–6.9 show the target and predicted values from the best-performing 5-model autoregressive models of Table 6.3, for ATL, DEN, and DFW datasets, respectively. It can be seen that the evolved ensemble models achieved a good fit for most of the in-sample and out-of-sample data range. Table 6.4 presents a gallery of good-generalising GP-AR(36) evolved models.

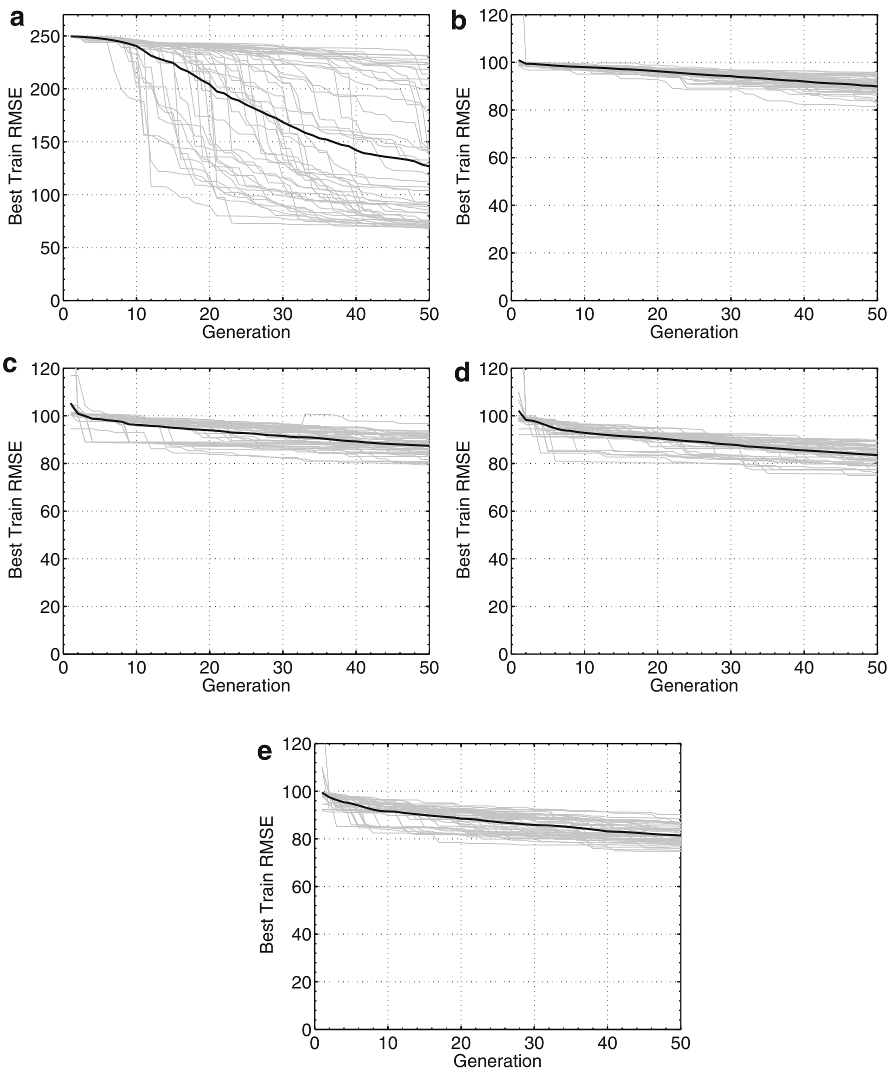


Fig. 6.6 RMSE histograms for the ATL dataset. Each figure illustrates 50 independent evolutionary runs. Average is indicated with *bold*. (**a**) GSR; (**b**) GP-AR(12); (**c**) GP-AR(24); (**d**) GP-AR(36); (**e**) GP-ARMA

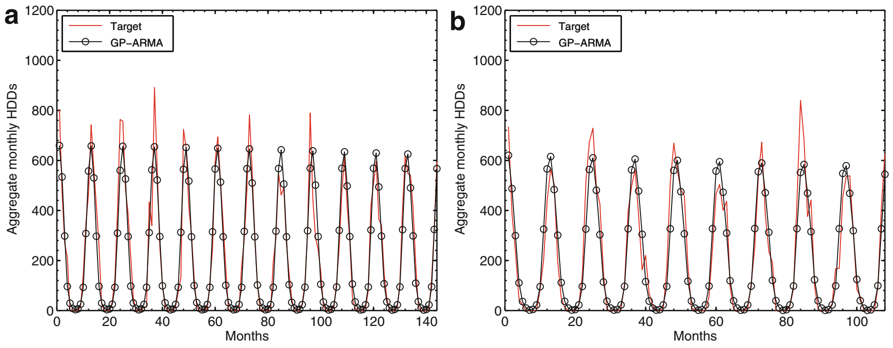


Fig. 6.7 Target vs. prediction for best-performing models of GP-ARMA (ensemble size 5) for the ATL dataset. (**a**) training data (**b**) test data

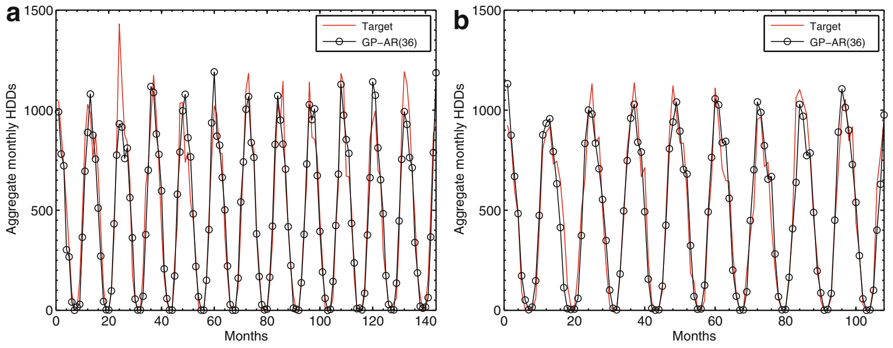


Fig. 6.8 Target vs. prediction for best-performing models of GP-AR(36) (ensemble size 5) for the DEN dataset. (**a**) training data (**b**) test data

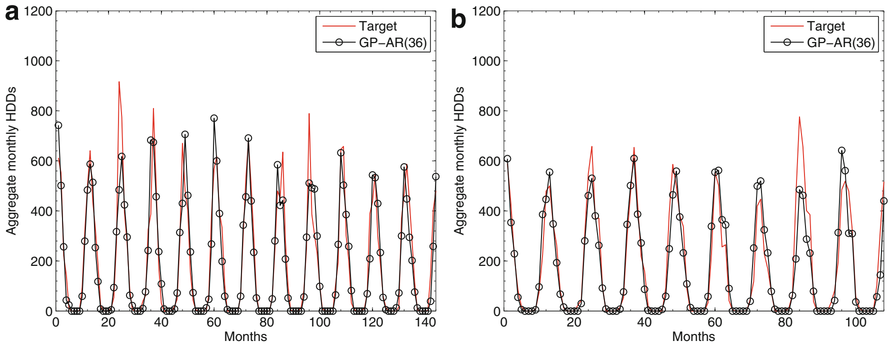


Fig. 6.9 Target vs. prediction for best-performing models of GP-AR(36) (ensemble size 5) for the DFW dataset. (**a**) training data (**b**) test data

Table 6.4 Sample evolved GP-AR(36) models

$f(t) = \sqrt{HDD_{t-12}\left[HDD_{t-36} + \sqrt{HDD_{t-12} * \left(\frac{HDD_{t-26}}{-0.92+(HDD_{t-7}*\log(HDD_{t-21}))}\right)}\right]}$
$f(t) = \sqrt{(HDD_{t-24}HDD_{t-36}) - HDD_{t-24} + 11.51}$
$f(t) = 0.94HDD_{t-36}$
$f(t) = HDD_{t-36} - \sqrt{HDD_{t-12}} + 0.41(HDD_{t-36} - HDD_{t-12})$
$f(t) = \frac{HDD_{t-36}}{\sqrt{\frac{HDD_{t-36}+5.11}{HDD_{t-12}}}}$
$f(t) = \sqrt{(HDD_{t-36} + 0.17)(HDD_{t-12} - 0.84)}$

6.8 Conclusion

This study adopted a time-series modelling approach to the production of a seasonal weather-metric forecast, as a constituent part of a general method for pricing weather derivatives. Two GP-based methods for time-series modelling were used; the first one is based on standard symbolic regression; the second one is based on autoregressive time-series modelling that is realised via an iterated single-step prediction process and a specially crafted terminal set of historical time-series values.

Results are very encouraging, suggesting that GP is able to successfully evolve accurate seasonal temperature forecasting models. The use of ensemble learning of 5-model predictors enhanced the generalisation ability of the system, as opposed to single-model predictions. Standard symbolic regression was seen to be able to capture the deterministic aspect of the modelled data and attained the best test performance; however, its overall performance was marked as unstable, producing some very poor-generalising models. On the other hand, the performance of search-based autoregressive and moving average models was deemed on average the most stable and best-performing in out-of-sample data.

Acknowledgements This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant Number 08/SRC/FM1389.

References

1. C. Tuite, A. Agapitos, M. O'Neill, A. Brabazon, *A Preliminary Investigation of Overfitting in Evolutionary Driven Model Induction: Implications for Financial Modelling. Applications of Evolutionary Computing, EvoApplications 2011: EvoCOMNET, EvoFIN, EvoHOT, EvoMUSART, EvoSTIM, EvoTRANSLOG*, vol. 6625, 2011 (Springer, Turin, 2011), pp. 121–130
2. A. Agapitos, M. O'Neill, A. Brabazon, T. Theodoridis, Maximum margin decision surfaces for increased generalisation in evolutionary decision tree learning, in *Proceedings of the 14th European Conference on Genetic Programming, EuroGP 2011*, vol. 6621, ed. by S. Silva, J.A. Foster, M. Nicolau, M. Giacobini, P. Machado (Springer, Turin, 2011), pp. 61–72
3. C. Tuite, A. Agapitos, M. O'Neill, A. Brabazon, Tackling overfitting in evolutionary-driven financial model induction, in *Natural Computing in Computational Finance (Volume 4), Volume 380 of Studies in Computational Intelligence*, Chap. 8, ed. by A. Brabazon, M. O'Neill, D. Maringer (Springer, New York, 2012), pp. 141–161
4. A. Kattan, A. Agapitos, R. Poli, Unsupervised problem decomposition using genetic programming, in *Proceedings of the 13th European Conference on Genetic Programming, EuroGP 2010*, vol. 6021, ed. by A.I. Esparcia-Alcazar, A. Ekart, S. Silva, S. Dignum, A. Sima Uyar (Springer, Istanbul, 2010), pp. 122–133
5. A. Agapitos, M. O'Neill, A. Brabazon, Evolutionary learning of technical trading rules without data-mining bias, in *PPSN 2010 11th International Conference on Parallel Problem Solving From Nature*, vol. 6238, ed. by R. Schaefer, C. Cotta, J. Kolodziej, G. Rudolph (Springer, Krakow, 2010), pp. 294–303
6. A. Agapitos, M. O'Neill, A. Brabazon, Stateful program representations for evolving technical trading rules, in *GECCO '11: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation* (ACM, Dublin, 2011), pp. 199–200
7. T. Theodoridis, A. Agapitos, H. Hu, A gaussian groundplan projection area model for evolving probabilistic classifiers, in *GECCO '11: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation* (ACM, Dublin, 2011), pp. 1339–1346
8. C. Tuite, A. Agapitos, M. O'Neill, A. Brabazon, Early stopping criteria to counteract overfitting in genetic programming, in *GECCO '11: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation* (ACM, Dublin, 2011), pp. 203–204
9. A. Agapitos, M. O'Neill, A. Brabazon, Promoting the generalisation of genetically induced trading rules, in *Proceedings of the 4th International Conference on Computational and Financial Econometrics CFE'10*, ed. by G. Kapetanios, O. Linton, M. McAleer, E. Ruiz (ERCIM, Senate House, University of London, UK, 2010), p. E678
10. A. Agapitos, M. Dyson, J. Kovalchuk, S.M. Lucas, On the genetic programming of time-series predictors for supply chain management, in *GECCO '08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation* (ACM, New York, 2008), pp. 1163–1170
11. P. Alaton, B. Djehiche, D. Stillberger, On modelling and pricing weather derivatives. Appl. Math. Finance **9**(1), 1–20 (2002)
12. M. Alvarez-Diaz, G. Caballero Miguez, M. Solino, The Institutional Determinants Of CO2 Emissions: A Computational Modelling Approach Using Artificial Neural Networks and Genetic Programming. FUNCAS Working Paper 401, Fundacion de las Cajas de Ahorros, Madrid, July 2008
13. M. Arganis, R. Val, J. Prats, K. Rodriguez, R. Dominguez, J. Dolz, Genetic programming and standardization in water temperature modelling. Adv. Civil Eng. (2009). Article ID 353960, doi:10.1155/2009/353960
14. A. Bakhshaii, R. Stull, Deterministic ensemble forecasts using gene-expression programming. Weather Forecast. **24**(5), 1431–1451 (2009)
15. C.M. Bishop, *Neural Networks for Pattern Recognition* (Oxford University Press, Oxford, 1996)

16. F. Black, M. Scholes, The pricing of options and corporate liabilities. J. Polit. Econ. **81**, 637–654 (1973)
17. T. Bollerslev, Generalised autoregressive conditional heteroskedasticity. J. Econometrics **31**, 307–327 (1986)
18. A. Brabazon, M. O'Neill, *Biologically Inspired Algorithms for Financial Modelling* (Springer, New York, 2006)
19. L. Breiman, Bagging predictors. Mach. Learn. **24**, 123–140 (1996)
20. S. Campbell, F. Diebold, Weather forecasting for weather derivatives. J. Am. Stat. Assoc. **100**(469), 6–16 (2005)
21. M. Cao, J. Wei, Equilibrium Valuation of Weather Derivatives. Working paper, School of Business, York University, Toronto (2002)
22. E. Carreno Jara, Long memory time series forecasting by using genetic programming. Genetic Programming and Evolvable Machines **12**(4), 429–456 (2012)
23. G. Considine, Introduction to weather derivatives. Technical report, Weather Derivatives Group (1999)
24. P. Coulibaly, Downscaling daily extreme temperatures with genetic programming. Geophys. Res. Lett. **31**, 1–4 (2004)
25. M. Davis, Pricing weather derivatives by marginal value. Quant. Finance **1**, 305–308 (2001)
26. I. De Falco, A. Della Cioppa, E. Tarantino, A genetic programming system for time series prediction and its application to El Nino forecast, in *Soft Computing: Methodologies and Applications*, ed. by F. Hoffmann, M. Köppen, F. Klawonn, R. Roy. Advances in Intelligent and Soft Computing, vol. 32 (Springer, Berlin, 2005), pp. 151–162
27. R. Duda, P. Hart, D. Stork, *Pattern Classification*, 2nd edn. (Wiley, New York, 2001)
28. B. Efron, R. Tibshirani, *An Introduction to the Bootstrap* (Chapman and Hall, New York, 1993)
29. J.J. Flores, M. Graff, E. Cadenas, Wind prediction using genetic algorithms and gene expression programming, in *Proceedings of the International Conference on Modelling and Simulation in the Enterprises. AMSE 2005*, Morelia, Mexico, April 2005
30. A. Garcia, F. Sturzenegger, Hedging corporate revenues with weather derivatives: A case study. Master's thesis, Universite de Lausanne (2001)
31. Y.-S. Hong, M.R. Rosen, Identification of an urban fractured-rock aquifer dynamics using an evolutionary self-organizing modelling. J. Hydrol. **259**(1–4), 89–104 (2002)
32. H. Iba, Bagging, boosting, and bloating in genetic programming, in *Proceedings of the Genetic and Evolutionary Computation Conference*, ed. by W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, R.R. Smith, vol. 2 (Morgan Kaufmann, San Francisco, 1999), pp. 1053–1060
33. S. Jewson, R. Caballero, The use of weather forecasts in the pricing of weather derivatives. Meteorol. Appl. **10**, 367–376 (2003)
34. S. Jewson, A. Brix, C. Ziehmann, *Weather Derivative Valuation: The Meteorological, Statistical, Financial and Mathematical Foundations* (Cambridge University Press, Cambridge, 2005)
35. O. Kisi, A. Guven, Evapotranspiration modeling using linear genetic programming technique. J. Irrigat. Drain. Eng. **136**(10), 715–723 (2010)
36. J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT, Cambridge, 1992)
37. J.R. Koza, Human-competitive results produced by genetic programming. Genetic Programming and Evolvable Machines **11**(3/4), 251–284 (2010)
38. A. Makkeasoyrn, N.-B. Chang, X. Zhou, Short-term streamflow forecasting with global climate change implications – A comparative study between genetic programming and neural network models. J. Hydrol. **352**(3–4), 336–354 (2008)
39. S. Makridakis, S. Wheelright, R. Hyndman, *Forcasting: Methods and Applications* (Wiley, New York, 1998)
40. M. Moreno, Riding the temp – Is it possible, in weather derivatives pricing models, to simulate the temperature effectively? Futures and Options World **15**, 22–26 (2001)
41. M. O'Neill, L. Vanneschi, S. Gustafson, W. Banzhaf, Open issues in genetic programming. Genetic Programming and Evolvable Machines **11**(3/4), 339–363 (2010)

42. R. Poli, W.B. Langdon, N.F. McPhee, *A Field Guide to Genetic Programming* (Lulu Enterprises, NC, 2008)
43. R. Poli, L. Vanneschi, W.B. Langdon, N.F. McPhee, Theoretical results in genetic programming: The next ten years? Genetic Programming and Evolvable Machines **11**(3/4), 285–320 (2010)
44. K. Rodriguez-Vazquez, Genetic programming in time series modelling: an application to meteorological data, in *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001* (IEEE Press, NJ, 2001), pp. 261–266
45. S. Shahid, M. Hasan, R.U. Mondal, Modeling monthly mean maximum temperature using genetic programming. Int. J. Soft Comput. **2**(5), 612–616 (2007)
46. J. Taylor, R. Buizza, Density forecasting for weather derivative pricing. Int. J. Forecast. **22**, 29–42 (2006)
47. J.J. Valdes, A. Pou, Central England temperatures and solar activity: a computational intelligence approach, in *International Joint Conference on Neural Networks (IJCNN 2010)* (IEEE Press, Piscataway, 2010), pp. 1–8
48. G.J.F.T. Van Sprundel, Using weather derivatives for the financial risk management of plant diseases: A study on Phytophthora infestans and Fusarium head blight. PhD thesis, Wageningen University (2011)
49. R. Vining, Weather derivatives: implications for Australia, in *Proceedings of Hawaii Conference on Business* (2001)
50. N. Wagner, Z. Michalewicz, M. Khouja, R.R. McGregor, Time series forecasting for dynamic environments: The DyFor genetic program model. IEEE Trans. Evol. Comput. **11**(4), 433–452 (2007)
51. Weather Risk Management Association. Results of 2006 annual industry-wide survey, April 2006
52. Weather Risk Management Association. Introduction to the weather market, April 2011
53. A. Weigel, D. Baggenstos, M. Liniger, Probabilistic verification of monthly temperature forecasts. Mon. Weather Rev. **136**, 5162–5182 (2008)
54. P.A. Whigham, P.F. Crapper, Time series modelling using genetic programming: an application to rainfall-runoff models, in *Advances in Genetic Programming 3*, ed. by L. Spector, W.B. Langdon, U.-M. O'Reilly, P.J. Angeline (MIT, Cambridge, 1999), pp. 89–104

Chapter 7
Evolution Strategies for IPO Underpricing Prediction

David Quintana, Cristobal Luque, Jose Maria Valls, and Pedro Isasi

Abstract The prediction of first-day returns of initial public offerings is a challenging task due, among other things, to an incomplete theory on the dynamics and the presence of outliers. In this chapter we introduce an evolutionary system based on prototypes adjusted by evolution strategies. The system, set up in two layers, breaks the input space into different regions and fits specialized sets of models. These models are then combined to offer predictions. The structure of the model is such that it is able to handle the extreme values that hinder prediction in this domain. The system is benchmarked against a set of well-known machine learning algorithms, and the results show competitive performance.

7.1 Introduction

The existence of abnormal first-day trading returns on initial public offerings (IPOs) is a phenomenon that has been documented in the academic literature for a long time. Historically, there has been a substantial difference between the price at which the stock is sold to investors and the closing price at the end of the first trading day. The size of gap between the price set by the sellers and their advisors and the market price, once the shares of the company have been trading freely, varies with time periods and industries and is difficult to justify from current financial theory. Ritter and Welch [42] report an average initial return of 18.8% on a sample of 6,249 US IPOs that took place between 1980 and 2001.[1]

[1]Equally weighted average first-day return measured from the offer price to the first CRSP-listed crossing price.

D. Quintana (✉) • C. Luque • J.M. Valls • P. Isasi
Universidad Carlos III de Madrid, Avda. Universidad, 30 Leganés 28911, Madrid, Spain
e-mail: dquintan@inf.uc3m.es; cluqueac@gmail.com; jvalls@inf.uc3m.es; isasi@ia.uc3m.es

There is a long history of research devoted to this issue. Researchers on financial economics have been trying to explain the phenomenon for decades and it has been the subject of a vast amount of academic work. Many theories have been postulated to offer explanations and it is still a very active field. The mentioned paper by Ritter and Welch is a good introduction to the domain.

The pricing of new stock is a complicated process that results from a negotiation process between the issuers and their financial advisors. Pricing entails building complicated models that deal with a wide array of uncertainties regarding the firm and their markets. Once an initial reference price is agreed, the company and the investment banks present the deal to potential investors during a process called the road show. These potential buyers, usually large institutional investors, show their interest making of nonbinding purchase offers. The financial advisors gather this new information regarding the potential demand and subsequently use it to adjust the original reference price and set the final offering price. The existence of information asymmetries, conflicting interests, uncertainties, and the state of financial markets leads to the mentioned first-day return pattern.

Misspricing could result in major economic inefficiency, specially in those cases where the difference is extremely large. An example that could be mentioned would be the case of Broadcom, a high-speed semiconductor products company. The company filed an offer price of \$24 and, by the end of the first day, the stock was trading at \$53.63. Depending on the point of view, underpricing could be seen as money left on the table or a considerable opportunity for those who can get shares allocated on the right IPOs. Given the sums involved, the ability to predict the initial return could be very profitable. Sellers could adjust the offering price to get more money for their shares, and buyers could improve their allocation of funds. Having said that, IPOs underpricing prediction pose a major challenge. The nature of the underpricing is not well understood, the set of variables identified as relevant in the literature is incomplete, and the presence of outliers adds difficulty to the task of predicting the initial performance.

Most of the empirical analysis that has been carried out to date in order to explain underpricing is based on linear models. These models lack the flexibility to capture nonlinearity and complex interaction among the independent variables. Despite these limitations, the efforts carried out using computational intelligence are very limited and are mostly based on artificial neural networks. They model it as a regression task and rely on multilayer perceptrons to solve it. The approach suggested in this chapter is very different as it is based on evolutionary system using a Michigan approach. The system evolves a set of Voroni regions using evolution strategies that are subsequently used to fit local models. These models, set up in a two layer structure, can then be used for prediction purposes.

The rest of the chapter is structured as follows: Sect. 7.2 introduces the main contributions to IPO underpricing prediction using computational intelligence. Section 7.3 presents the explanatory variables and describe the data used in the empirical analysis. Section 7.4 describes the evolutionary system and Sect. 7.5 will be used to report the results of the empirical analysis. Finally, Sect. 7.6 covers the summary and conclusions.

7.2 Computational Intelligence Literature in IPO Underpricing Prediction

The use of computational intelligence in financial prediction is hardly new. Many pieces of research have explored the suitability of this technique in areas such as trading [4, 16, 17], portfolio optimization [12, 26, 28], or bankruptcy prediction [5, 34, 46] but little has been done in IPO research. Given the nature of the problem, it has been traditionally tackled using artificial neural networks.

Jain and Nag [24] conducted the first attempt to predict IPO underpricing. In their paper, they train a set of multilayer perceptrons trained with backpropagation to predict the first-day close of a set of US IPOs. They used a set of 11 input variables representing different financial indicators traditionally used by investors. The results generated by the networks were closer to the offering prices set by the issuers and investment bankers, leading the authors to conclude that neural networks outperform issuers and investment bankers in pricing IPOs. Their work was closely followed by those of Coy et al. [15] and Robertson et al. [43], who modeled the problem in a very similar way. The authors of the latter extended the number of predictors to 16 and targeted the initial return. Their main contributions was segmenting the data into two samples, technology and not technology, and adding factorial analysis to understand the effects of different parameter settings. They also report that the neural networks trained with backpropagation beat ordinary least squares (OLS) regression in both samples.

Another related paper is the one presented by Mitsdorffere et al. [36]. Unlike the previous authors, these researchers did not target the specific underpricing. Instead, they modeled the problem as a classification task and tried to identify IPOs that showed first-day price run-ups over 50%. In order to do that, they tested a set of machine learning methods (Bayesian classifications, support vector machines, decision tree techniques, rule, learners and artificial neural networks) on a sample of 182 US tech IPOs. They concluded that C4.5, followed by neural networks, offered the best results.

More recently, other authors have tested these models on non-US samples. The second most studied IPO market using computational intelligence, after the US, has been the Chinese one. This market is characterized by abnormally high initial returns. Tian [49], for instance, reports an average underpricing of 267% between 1991 and 2001 and explains this extraordinary performance through a combination of specific regulations and investment risks. Meng [35] tested artificial neural networks on a sample of 99 Chinese SME IPOs preprocessing the data. He used principal component analysis to reduce the dimension of the input space from 18 variables to 6 factors. Chen and Wu [11] also tested a set of multilayer perceptrons on Chinese SME IPOs to predict first day's closing prices. Finally, Chou et al. [13] use a genetic algorithm to optimize the architecture of the neural network but, apart from that, there are no major differences compared to the mentioned studies.

Reber et al. [40] compare linear regression and neural network models on a UK sample and obtain the best performance using simple neural networks based on seven inputs. They also report instability of the models across different partitions of the data. However, it is important to note that this phenomenon was observed across the board. It was not something specific of neural networks.

The alternatives to neural networks have been very limited. Quintana et al. [39] introduced a system that evolves a set of prediction rules using a steady-state genetic algorithm with a Michigan approach. Once the system is trained, it can then be used to predict underpricing for new IPOs matching relevant rules to the values of seven independent variables identified in the financial literature. The authors compared its performance to OLS regression and robust regression models and obtained the best results with the evolutionary system. Luque et al. subsequently introduced another method based on evolution strategies in an earlier version of this work [32] offering competitive results.

7.3 Variables and Data

In this section, we introduce formally the phenomenon we have been referring to as IPO underpricing . Then, we will describe the set of independent variables that will be used in the models for prediction purposes.

We define IPO underpricing as the percentage change of the share price from the offer to the closing price on first day of trading minus the return on the appropriate index or:

$$R_i = \left(\frac{Pc_i - Po_i}{Po_i}\right) - \left(\frac{Mc_i - Mo_i}{Mo_i}\right),$$

where R_i is the adjusted first-day return for stock i; Po_i is the offering price for stock i; Pc_i is the closing price for stock i; Mo_i is the closing for the broad market index of the market where the stock i was taken public for the day before the IPO and Mc_i is the closing for the broad market index of the market where the stock i was floated on the day of the IPO.[2]

Once the target variable has been clearly described, we introduce the independent variables.

7.3.1 *Explanatory Variables*

As we have already mentioned in the introduction, the amount of literature regarding IPO underpricing is quite remarkable. This fact makes the initial number of potential explanatory variables very high. However, there seems to be a number of them concerning the structure of the offerings that show up often in academic studies.

[2]The indexes used in the analysis were: S&P 500, NASDAQ Composite and AMEX Composite.

These variables, which are about to be succinctly described, are the following: price range width, price adjustment, offer price, retained stock offer size, and relation to tech sector.

7.3.1.1 Price Range Width (RANGE)

This variable represents the width of the nonbinding reference price range offered to potential customers during the roadshow. This width can be interpreted as a sign of uncertainty regarding the real value of the company and, therefore, as a factor that could influence the initial return. Following [21, 25], the representation to be used will be the difference between the maximum and minimum price divided by the minimum price.

7.3.1.2 Price Adjustment (P_ADJ)

Authors of [8, 21, 29] suggest the relation between the final offer price and the mentioned price range might also be interpreted as sign of uncertainty. They state that this effect might be captured by the following expression:

$$P\_ADJ = \frac{P_{\mathrm{f}} - P_{\mathrm{e}}}{P_{\mathrm{e}}},$$

where P_{f} is the final offer price and P_{e} is the expected price defined as the middle point of the price range.

7.3.1.3 Offering Price (PRICE)

The final offering price has been found to be a relevant variable not only as a part of the previous indicator but also on its own. Studies like [3, 9, 10], among others, support this idea.

7.3.1.4 Retained Stock (RETAINED)

The influence of the capital retained by initial investors at the time of the IPO has been traditionally understood to signal the quality of the stock [1, 18, 27, 29]. Since we lack the breakdown of primary and secondary shares, we will proxy this variable through the ratio number of shares sold at the IPO divided by post-offering number of shares minus the number of shares sold at the IPO.

Table 7.1 Descriptive statistics

	Mean	Median	Max	Min	Std. dev.
RETURN	0.176	0.073	3.718	−0.281	0.389
PRICE	14.641	14.000	85.000	3.250	5.812
RANGE	0.149	0.143	0.500	0.000	0.063
P_ADJ	0.105	0.082	1.509	0.000	0.099
LSIZE	2.054	2.025	3.939	0.061	0.446
RETAINED	0.309	0.262	1.000	0.002	0.198
TECH	0.328	0.000	1.000	0.000	0.470

7.3.1.5 Offering Size (LSIZE)

This variable is defined as the logarithm of the offering size in millions of dollars excluding the over-allotment option. Studies like [3, 22, 23, 33] support the need to include it in the models.

7.3.1.6 Technology (TECH)

The reason why we suggest a specific variable to control whether the industrial activities of a company are related to tech sector is the fact that they tend to show a higher underpricing. This fact is usually modeled by a dummy that equals one for tech companies [30, 31, 47]. Our labeling criterion is based on IPO Monitor's definition. This company publishes reports with the list of tech companies taken public based on US Standard Industry Codes. Hence, we will consider an IPO to be "Tech" if it is in the list.

7.3.2 Data

The sample used in the empirical section of this work consisted of 866 companies taken public between January 1999 and May 2010 in US stock markets. This includes AMEX, NASDAQ, and NYSE IPOs and excludes American Depositary Receipts; closed-end funds; real-state investment trusts and unit offerings. Our primary data source was IPO Monitor. The information was completed with IPO profiles from Hoovers. Index information was obtained from NASD (NASDAQ and AMEX composites) and DataStream (S&P 500).

The main descriptive statistics (mean, median, maximum, minimum and standard deviation) for the sample of 866 IPOs considered in the analysis are reported in Table 7.1.

7.4 Description of the Evolutionary System

In this chapter, we present a new supervised learning system based on evolution strategies [41, 45] to perform a prediction tasks. Usually, this kind of systems relies on the whole training data set to derive the rules used to make predictions. However, in some domains, the peculiarities of the input space favor approaches that stress the importance of local information. Among these, we could mention the existence of significant differences among the different regions of the input space. The system introduced in this work tries to automatically detect the areas in the input space that share enough features to be accurately predicted by the same set of rules. In addition to that, the system will determine which rules are appropriate for each of them. Thus, at the end, it is going to produce on one hand a partition of the input space in different areas by means of a set of prototypes and the nearest neighborhood rule, and, on the other hand, a set of rules for each region that defines the predictions that are going to be generated for that region.

As in most machine learning algorithms, two stages are required: a training stage, to create a model, and a test stage, to validate the model. As part of the training stage, the input space is divided into Voronoi regions by means of a set of prototypes. In each of these Voronoi regions, a linear regression is performed to fit the points representing the instances of the training set, defined by pairs (input, output). Thus, several local linear regressions are performed, one per region.

Once the training process is finished, given a test pattern, the system will assign it to the appropriate region, following the nearest neighborhood rule, and then it will provide a prediction based on the local regression associated to that region. The algorithm has been conceived in a way that tends to allocate noisy patterns into specific regions, hence improving the accuracy of the models fitted to the patterns that show a clearer structure. Next, we will describe the learning procedure and how the system works.

In this work we use the previously defined set of independent variables (LSIZE, P_ADJ, PRICE, RANGE, RETAINED and TECH) and RETURN as target variable.

7.4.1 *Training Process*

The training process divides the input variable space into Voronoi regions. Let n be the dimension of the input space, and thus $n+1$ the dimension of the pattern space, where the extra dimension represents the prediction. We will call "prototype" to a point P (a vector) in the input variable space (in this particular domain, R^6). Given a set of k prototypes $P_i \in R^n$, where $i \leq k$, and n is the dimension of the input space, this set divides R^n into k Voronoi regions, when the nearest neighborhood rule is used. If we call V_i to the region defined by P_i, we can describe this region mathematically as

$$V_i := \{P \in R^n / d(P, P_i) < d(P, P_j) \text{ for all } j \leq k \text{ such that } j \neq i\},$$

where d is the standard Euclidean distance.

The goal of this phase is to determine the best partitioning in terms of predictive accuracy. The approach suggested to do this is based on an evolutionary process. Each prototype is represented by an individual in an evolutionary strategies system. The learning procedure of the evolutionary strategies moves those prototypes to place them in the location where the predictive properties are maximized. The predictive properties of the regions are used as the fitness value of the individuals.

7.4.1.1 Evaluation of Fitness

Training patterns are composed of vectors of $(n+1)$ dimensions (n dimensions for the input variables and one dimension for the output). For the encoding of the individuals, just the n first dimensions, corresponding to the n input variables, are used. Let $T = (t_1,\ldots,t_{n-1},t_n,t_{n+1})$ be a training pattern. A projection map from R^{n+1} to R^n has to be defined:

$$\Pi(t_1,\ldots,t_{n-1},t_n,t_{n+1}) := (t_1,\ldots,t_{n-1},t_n).$$

Those projections are used as training elements for the generation of the Voronoi regions. In order to compute the fitness value of an individual (prototype P_i), we need first to assign a set of pattern projections to that prototype, following the nearest neighborhood rule:

$$\Pi(T) \in V_i \iff d(T,P_i) < d(T,P_j) \text{ for all } j \leq k \text{ such that } j \leq i.$$

Once every training pattern is assigned to its corresponding region, a regression R_i is calculated for each region V_i. R_i is a linear regression of the output variable over the input variables for each pattern T, such that $\Pi(T) \in V_i$. Let $R_i(T)$ be the estimated output for the pattern T by the regression R_i, and T_0 the output value of the pattern T. That is, if $T = (t_1,\ldots,t_n,t_{n+1})$, then $T_0 := t_{n+1}$. Then, the error of that estimation is $E = |T_0 - R_i(T)|$. Thus, we have the relationship $P_i \to V_i \to R_i$ for each $i \leq k$. That is, a prototype P_i defines a region V_i, and that region has an associated regression R_i.

The goal of the algorithm is to minimize the sum of errors. In standard evolutionary algorithms, a fitness value is assigned to each individual. For instance, we could use the sum of the errors for all the patterns associated to the region V_i as the fitness function for each individual (prototype P_i). However, we consider that it would not be suitable in this setting since the individuals with fewer associated patterns (that is, fewer projections in their region, and therefore, a sum of less error terms) would tend to behave better. Another candidate for fitness function could be the previous one divided by the number of patterns belonging to the region associated to the individual (mean error). This choice would not be much better than the previous one, because it would allow individuals with more patterns in their regions to evolve stealing patterns from the nearby regions, and thus, the error derived from using a regression that is not suitable would be compensated by the

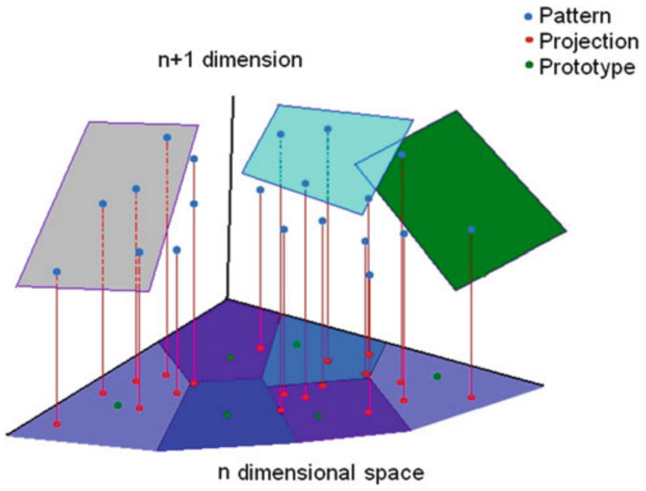


Fig. 7.1 Graphical representation of the way the input space is broken down into Voronoi regions

fact that the magnitude of such error would be shared among all the data points in the region. The dilution of the consequences of the mentioned undesired behavior has led us to look for another alternative, a population-based fitness (Michigan approach). That means that instead of having a fitness value for each individual, we have a fitness value for all the population. This value is the sum of the errors $E = |T_0 - R_i(T)|$ for each pattern T and i such that $\Pi(T) \in V_i$:

$$\text{Fitness} = \sum_T |T_0 - R_i(T)|, \; i \text{ such that } \Pi(T) \in V_i.$$

This approach is illustrated in Fig. 7.1.

7.4.1.2 Evolution Process

The initial population is randomly created. The evolution process basically works as a $(1+1)$ parallel evolution strategy. That is, in every generation, each parent produces an offspring by mutation. Let $I = (x_1, \ldots, x_n, \sigma_I)$ be an individual and let $I' = (x'_1, \ldots, x'_n, \sigma'_{I'})$ be its offspring. The mutation process can be expressed as

$$x'_i = N(x_i, \sigma_I),$$

where $N(X,Y)$ represents a normal random variable with mean X and variance Y, and σ_I is the variance of the parent. In order to mutate the offspring's variance $\sigma'_{I'}$, the offspring carries a counter for the last successful replacement of his parent by its offspring. That is done using the $1/5$ rule suggested by Rechenberg, to adapt the variance of the mutations I, as explained in [7].

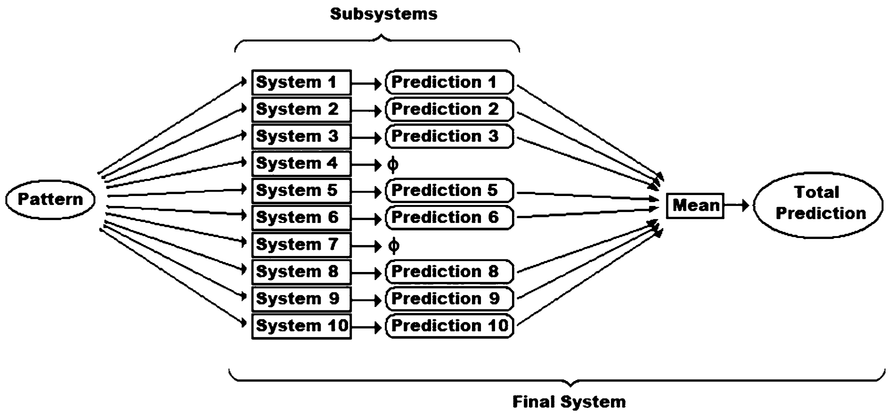


Fig. 7.2 Schema of the two phases of prediction of the evolutive system

For each offspring I_0 we look for the closest individual I in the population in terms of Euclidean distance. Both individuals must be compared in order to select the best, so we calculate the population fitness before and after replacing I by I_0. Finally, if the population fitness is worse after the replacement, we undo the change and keep the population as it was before.

7.4.2 Prediction Process

This evolutionary approach allows us to build a predictive model composed of several local linear regressions associated to the specific regions.

With the aim of improving the quality of the predictions, instead of one, ten models or subsystems with different random initializations will be built, in order to make the predictions more reliable and accurate. The reason behind this is that a regression is able to produce a reliable prediction only if the number of points it has been built with is big enough. Because of this, the regions composed of a scarce number of points will not be allowed to produce an output.

Therefore, our system is composed of ten subsystems, as it can be seen in Fig. 7.2. Each one is a prediction model randomly initialized and evolved with the training patterns, as it has been explained above. Some subsystems might not produce a valid output because the corresponding Voronoi regions are not reliable enough. However, the final output of the whole system is the mean of the valid—and reliable—outputs of the other subsystems. The whole system is now likely to produce reliable predictions for new testing patterns.

Let us summarize the prediction process: For a given testing pattern T, each subsystem must establish, in the first place, the region in which the pattern is located. Then, the prediction is calculated as the estimated value of T by the regression that corresponds to that region.

Algorithm 1 Prediction Algorithm

```
STEP 1: find $i$ such that $\Pi(T) \in V_i$
STEP 2: if ($\sharp Ri$ >MIN) then
            OUTPUT:= $R_i(T)$
        else NO-OUTPUT
```

The prediction process for each subsystem could be summarized as in Algorithm 1, where $\sharp R_i$ is the number of points in the region R_i after the training process and MIN is the minimum of points required for a prediction. The MIN parameter depends on the accuracy we desire for the algorithm. For linear regressions, in order to make its prediction reliable, it is usually recommended to be calculated with, at least, 5 points for each variable. Therefore, the number of input variables multiplied by 5 is a good value for MIN. For the experiments done, the algorithm has been tested with 3, 5, and 10 patterns per input variable for each region as condition in order to produce an output.

For a given pattern, each subsystem may produce an output or not. The final output of the system for a pattern T is the mean of the valid values returned by each subsystem, as Fig. 7.2 shows.

7.5 Empirical Analysis

The aim of this chapter is to show an evolutionary algorithm that could be very useful in IPO research. We will illustrate its potential comparing its forecasting ability to the one achieved by several and widely used machine learning alternatives. This comparison will be based on a 100-fold cross-validation analysis using the 866 patterns introduced before. The accuracy of the predictions will be assessed comparing the root mean square error (RMSE) of the predictions of the models. We will apply all the algorithms to the same data sets in order to get comparable results.

The data was normalized in the [0,1] interval and subsequently used to fit the described model. We did 100 experiments with 3, 6, 12, and 24 individuals, and the results are reported in Table 7.2. Each row shows the average of the results for each number of individuals. Column "RMSE" shows the Root Mean Square Error, column "Var." displays the variance of the RMS Error, and column "Pred. %" the percentage of patterns in the validation set for whom the system could produce a reliable prediction. As it was explained before, the system has the ability to anticipate which observations could lead to potentially inaccurate forecasts. Should this be the case, the system would not provide any estimation. The percentage of the sample predicted depends on the minimum number or points that a region is required to have. This parameter is set in function of the number of input variables in the data set, in this case, six, and the number of patterns per variable required to fit a reliable regression model. Table 7.2 shows the results for this minimum number of points value required, when we set the minimum number of patterns per variable

Table 7.2 Results of the experiments for the Voronoi system

	18 p./reg.			30 p./reg.			60 p./reg.		
Indiv.	RMSE	Var.	Pred.%	RMSE	Var.	Pred.%	RMSE	Var.	Pred.%
3	0.0802	0.00239	99.8	0.0796	0.00247	99.8	0.0768	0.00244	99.8
6	0.0763	0.00223	99.8	0.0757	0.00220	99.8	0.0751	0.00223	99.7
12	0.0745	0.00232	99.6	0.0744	0.00230	99.6	0.0728	0.00219	99.4
24	0.0705	0.00202	99.0	0.0700	0.00191	98.6	0.0689	0.00180	97.6

to 3, 5, and 10. This means that we will test the system with requirements of at least 18, 30 and 60 points for region. The reported figures confirm that the larger this number is, the higher predictive accuracy.

The best result in terms of prediction error, 0.0689, is achieved using both the largest number of regions and the largest minimum number of patterns per region required to make predictions. Ceteris paribus, Larger values in any of the mentioned two parameters result in both lower prediction errors, and lower prediction rates. This makes sense as one of them results in more specialization and the other in linear models fitted on more data. Combined, they result in more areas potentially devoted to outliers that will not meet the requirement to be predicted and, therefore, will not drag the results.

As we mentioned before, we repeated the same cross-validation analysis using a set of machine learning alternatives. For such a task, we used a powerful, well-known and widely used Java package called Waikato Environment for Knowledge Analysis (WEKA) [20]. The algorithms we will use as benchmarks are conjunctive rules, IBK, K∗, LWL, M5 Rules, M5P, radial basis neural networks, multilayer perceptron neural networks, and SMO for regression.

- Conjunctive rule [20]: Single conjunctive rule learner.
- IBK [2]: K-nearest neighbor classifier algorithm.
- K∗ [14]: Is an instance-based classifier, that is the class of a test instance is based upon the class of those training instances similar to it, as determined by some similarity function.
- LWL [6]: This is a local instance-based weighted learning algorithm. It builds a classifier from the weighted instances.
- M5Rules [19]: Generates a decision list for regression problems using separate-and-conquer.
- M5p [38]: Numerical classifier that combines a conventional decision tree with the possibility of linear regression to predict continual variables.
- Multilayer Perceptron: Algorithm that simulates the biological process of learning through weight adjusting using backpropagation algorithm [44].
- RBNN [37]: Radial basis neural networks are another type of artificial neural network. It uses radial basis functions to approximate different regions of the input space depending on their characteristics.
- SMO-Reg [41]: Implements sequential minimal optimization algorithm for training a support vector regression using polynomial or RBF kernels.

Table 7.3 Results of the experiments for other algorithms implemented in WEKA

Algorithm	RMSE	Variance
Conj. rule	0.0849	0.002833
IBK	0.0767	0.001979
K*	0.0780	0.001854
LWL	0.0834	0.002729
M5P	0.0830	0.001983
M5Rules	0.0850	0.002060
Percep	0.0948	0.002326
RBNN	0.0813	0.002710
SMO-Reg	0.0766	0.003046

Table 7.4 Significance tests for the average prediction error

	Vor 3R 60P	Vor 6R 60P	Vor 12R 60P	Vor 24R 60P
Voronoi 6 Reg 10 Pt	=			
Voronoi 12 Reg 10 Pt	=	=		
Voronoi 24 Reg 10 Pt	=	=	=	
Conj. rule	++	++	++	++
IBK	=	=	+	++
K∗	=	+	++	++
LWL	++	++	++	++
M5P	++	++	++	++
M5Rules	++	++	++	++
Percep	++	++	++	++
RBNN	+	+	++	++
SMO	=	=	=	=

The mean errors and variance of the experiments using the default parameters are displayed in Table 7.3.

The best alternative seems to be SMO-Reg, closely followed by IBK and K∗. These perform at a level close to the algorithm using three regions and a 60-pattern per region requirement. In that case, the evolutionary algorithm offers a slightly higher RMSE for a percentage prediction of 99.8%. For any other case, the Voronoi System beats any alternative. The significance of the difference of the prediction errors was formally tested using Mann–Whitney test. The results of these tests are reported in Table 7.4. There, we represent the fact that the element in the row has an average prediction error significantly larger than the one in the column at 1% by "++," and the opposite by "−." In case it is significantly larger at 5% we will use "+." Finally, if we cannot discard at 5% the possibility of equal accuracy, we report "=." Table 7.4 shows the results of these tests.

Table 7.4 shows that the Voronoi system with 12 or 24 regions and at least 60 patterns per region outperforms significantly any alternative but SMO. With 24 regions, all these differences are significant at the 1% conventional level. Regarding the support vector machine, even though the prediction error is systematically higher for any combination of regions and minimum number of patterns, when the number

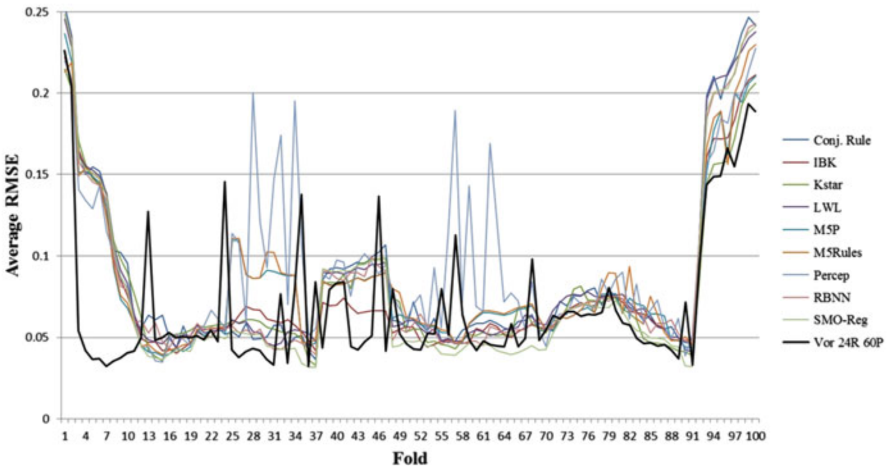


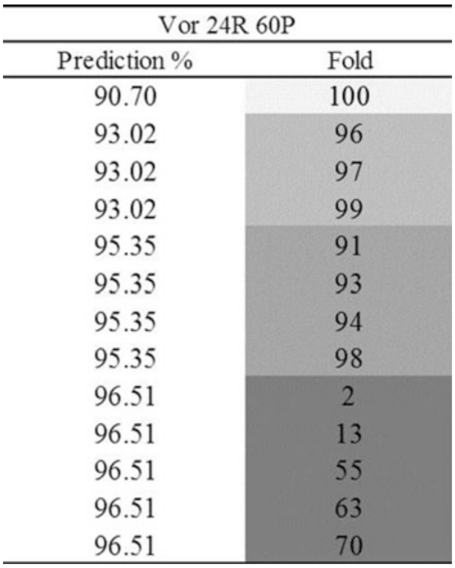
Fig. 7.3 Average root mean squared prediction error per fold and prediction method (the *darkest line* represents the evolutionary system)

of regions is higher than 3, the difference is not large enough to reject the null hypothesis of equality. The performance of the evolutionary algorithm is statistically identical across the different combinations of parameters.

In order to get a better understanding of the reason why the evolutionary system outperforms the alternatives, we will analyze the results by folds. Figure 7.3 shows the average RMSE for the configuration that offered the lowest prediction error, 24 regions and a minimum requirement of 60 patterns per region, and the machine learning alternatives across the 100-folds. The darkest line represents the evolutionary method.

The graph shows some of the IPOs are harder to predict than others. As it was mentioned before, the presence of outliers is a major factor in this domain. In this case, the patterns predicted in the first and the last few folds seem to be specially problematic, together with the mid-twenties and mid-forties. However, the largest contribution to the prediction error of most of the algorithms comes from the first set. The average prediction error for the algorithms rarely goes beyond the 0.15 threshold. The only exception is the multilayer perceptron. This model tends to show a performance that is in line with the rest, but seems to be more prone to large errors in specific cases.

The approach based on evolution strategies shows an interesting behavior. Most of the time, the prediction error is below average. However, we also observe spikes between 0.1 and 0.15 with higher than average relative frequency. In spite of this, the prediction error is still the lowest among the alternatives. The chart suggests that the explanation might be related to the extreme errors. For some specific folds, the average prediction errors tend to be extremely high. This is likely to be caused by the presence of outliers, that is, IPOs whose initial returns are hard to anticipate using the suggested set of independent variables. This is very relevant because the relative



Vor 24R 60P	
Prediction %	Fold
90.70	100
93.02	96
93.02	97
93.02	99
95.35	91
95.35	93
95.35	94
95.35	98
96.51	2
96.51	13
96.51	55
96.51	63
96.51	70

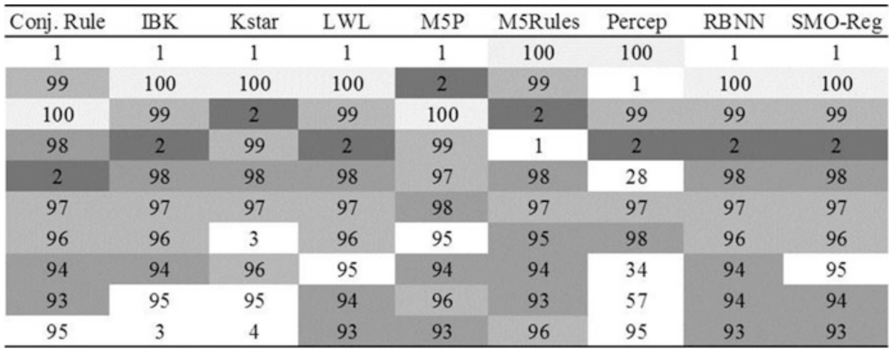
Fig. 7.4 Folds with the lowest prediction rates for the Voronoi system with 24 regions and a minimum of 60 patterns per region

magnitude of the errors for these folds is so large that it makes a major contribution to the total average RMSEs. Interestingly, in these folds, the evolutionary system tends to make predictions whose average accuracy is significantly higher than the rest. In fact, the difference seems to be so large in these cases that it more than compensates the mentioned spikes on regular patterns. The source of this advantage is likely to be found on the feature that was introduced before, more specifically, the one that prevents the system from making any forecast whenever it is anticipated that such prediction is likely to be unreliable.

As we mentioned on the previous section, the system requires a minimum number of patterns per region to make predictions. In case the minimum is not met, the regressions are considered unreliable. This feature, together with the way the prototypes are evolved, results on the isolation of some problematic areas of the input space. Further analysis of the prediction errors broken down by fold will illustrate the connection between this and the reported performance.

Figure 7.4 shows the folds with the 13 lowest prediction rates by the evolutionary algorithm. There, we can see that fold 100 got predictions for 90.70% of the patterns considered, folds 96, 97, and 99 got 93.02%, and so on. This information has been color-coded and put in relation with the prediction errors of the machine learning algorithms.

Figure 7.5 reports the ten highest average prediction errors by fold and alternative. The shading highlights the overlap between the difficulty anticipated by the system based on evolution strategies and the actual prediction errors. As it is apparent, the system introduced in this chapter correctly anticipates a large portion of the difficulties found by the competing algorithms. The degree of coincidence ranges from 50 to 90%, but goes up to 70–80% for the best ones. The color code



Conj. Rule	IBK	Kstar	LWL	M5P	M5Rules	Percep	RBNN	SMO-Reg
1	1	1	1	1	100	100	1	1
99	100	100	100	2	99	1	100	100
100	99	2	99	100	2	99	99	99
98	2	99	2	99	1	2	2	2
2	98	98	98	97	98	28	98	98
97	97	97	97	98	97	97	97	97
96	96	3	96	95	95	98	96	96
94	94	96	95	94	94	34	94	95
93	95	95	94	96	93	57	94	94
95	3	4	93	93	96	95	93	93

Fig. 7.5 Top ten highest average prediction errors by fold for the machine learning algorithms

considers the 13 lowest prediction percentages, not 10, due to the fact that there are fivefolds with a prediction rate of 96.51%. The system, however, seems to have been unable to identify fold 1 as specially problematic while, at the same time, every single algorithm shows extremely high average RMSE (the highest for 7 out of 9 machine learning algorithms). We could say something similar about fold 95, which is not among the top 13 lowest prediction rates for the evolutionary system in spite of being between the 7th and 10th hardest for 7 out of the mentioned 9.

It is worth noting the connection between the forecasting errors of the system and identification of outliers. As we can see in Fig. 7.3, fold 1 had the highest average RMSE and 95 was also among the most difficult to predict. Even though folds 96–100 were among the least predicted, RMSEs are very high. However, in spite of this being the case, the errors are clearly lower than average compared to the alternatives. The success of the outlier isolation behavior is specially remarkable in the first few folds, which explain to a large degree the differences in the global performance reported in Tables 7.2–7.4.

7.6 Summary and Conclusions

In this chapter we introduced an implementation of two-layer system based on evolution strategies to predict the underpricing of initial public offerings (IPOs).

The system divides the input variable space into Voronoi regions, which cover a set of patterns that are expected to behave in a similar way, and fits linear regressions that provide local predictions. The mentioned regions are defined by a set of nearest-neighbor prototypes whose location is updated using an evolution strategy with a Michigan approach. The process is run in parallel several times, each of them resulting in a component of a first prediction layer for the system. The outputs of these models are subsequently combined by a second layer to produce a single prediction.

This domain is particularly complicated due to the presence of outliers. Fortunately, the suggested approach is specially suitable as its design includes a specific mechanism to deal with this problem. The solution is based on a requirement to produce outputs. The system assesses the reliability of its predictions and only provides a forecast when it is expected to be accurate. The maximum allowed degree of uncertainty is set specifying a minimum number of patterns per Voronoi region. Below that level, prediction model is considered unreliable and the system does not provide forecasts for the region. This, combined with the way prototypes are evolved, results in specialized regions that both identify outliers among new patterns and reduce the distortion they introduce in the training sample.

The algorithm was tested on a sample of US IPOs using a 100-fold cross validation. The predictions were based on six variables identified by literature review and their accuracy has been compared to the forecasts provided by a set of machine learning algorithms.

The system was compared in terms of prediction error to nine alternatives implemented in WEKA: conjunctive rules, IBK, K*, LWL, M5P, M5Rules, multilayer perceptrons, RBNN and SMO for regression. The experimental results show statistically significant differences in favor of evolutionary system for all cases but SMO. The highest average prediction error of the Voronoi system for the worst-case scenario (3 regions and 18 patterns) was lower than the one obtained by six out of nine alternatives. That configuration offers a 98.5% prediction rate. This percentage goes down to 97.6% for 24 regions and a minimum of 60 patterns per region, which is the configuration with the lowest average root mean squared error.

The analysis of the results by fold shows that the difficulty of predicting underpricing is variable. However, most of the machine learning algorithms show the highest average prediction error in the same folds. Interestingly, the evolutionary system refrains to a larger degree from making predictions in most of those folds. This shows that the rule system is identifying outliers in the right places. Even though the forecasting error from the evolutionary system might be still high in these folds, it tends to be low in relative terms when we compare it to the rest of alternatives.

All the above suggests that IPO research would benefit from the use of this tool. We understand that so would prediction efforts in other domains where the presence of outliers is a relevant factor.

There are several potential lines of research that could extend the work presented here. Among them, we could mention the development of strategies that might enable higher prediction percentages, minimizing the cost in terms of prediction error. A second alternative that remains open is the possibility of developing hybrid solutions that combine the advantages of good function approximators with the outlier management capabilities of the evolutionary system.

Acknowledgements The authors acknowledge the financial support granted by the Spanish Ministry of Science and Technology under contract TIN2008-06491-C04-03 (MSTAR) and Comunidad de Madrid (CCG10-UC3M/TIC-5029).

References

1. R.K. Aggarwal, L. Krigman, K. Womack, Strategic IPO underpricing, information momentum, and lockup expiration selling. J. Financ. Econ. **66**(1), 105–137 (2002)
2. D. Aha, D. Kibler, Instance-based learning algorithms. Mach. Learn. **6**, 37–66 (1991)
3. S.M. Albring, R.J. Elder, J. Zhou, IPO underpricing and audit quality differentiation within non-big 5 firms. Int. J. Audit. **11**, 115–131 (2007)
4. F. Allen, R. Karjalainen, Using genetic algorithms to find technical trading rules. J. Financ. Econ. **51**, 245–272 (1999)
5. A. Atiya, Bankruptcy prediction for credit risk using neural networks: A survey and new results. IEEE Trans. Neural Network **12**(4), 929–935 (2001)
6. C.G. Atkeson, A.W. Moore, S. Schaal, Locally weighted learning. Artif. Intell. Rev. **11**, 11–73 (1997)
7. T. Bäck, H.P. Schwefel, Evolutionary algorithms: some very old strategies for optimization and adaptation, in *New Computing Techniques in Physics Research II*, ed. by D. Perret-Gallix (World Scientific, Singapore, 1992), pp. 247–254
8. L.M. Benveniste, P.A. Spindt, How investment bankers determine the offer price and allocation of new issues. J. Financ. Econ. **24**, 343–362 (1989)
9. M. Brennan, P. Hughes, Stock prices and the supply of information. J. Finance **46**, 1665–1691 (1991)
10. A. Chalk, J. Peavy, Initial public offerings, daily returns, offering types and the price effect. Financ. Analysts J. **43**, 65–69 (1987)
11. X. Chen, Y. Wu, IPO pricing of SME based on artificial neural network, in *Proceedings of the 2009 International Conference on Business Intelligence and Financial Engineering* (IEEE Computer Society, Los Alamitos, 2009), pp. 21–24
12. Y. Chen, S. Mabu, K. Hirasawa, Genetic relation algorithm with guided mutation for the large-scale portfolio optimization. Expert Syst. Appl. **38**(4), 3353–3363 (2011)
13. S. Chou, N. Yen-Sen, T.L. William, Forecasting IPO price using GA and ANN simulation, in *Proceedings of the 10th WSEAS International Conference on Signal Processing, Computational Geometry and Artificial Vision* (World Scientific and Engineering Academy and Society (WSEAS), WI, 2010), pp. 145–150
14. J.G. Cleary, L.E. Trigg, K*: an instance-based learner using an entropic distance measure, in *Proceedings of the 12th International Conference on Machine Learning*, ed. by A. Prieditis, S.J. Russell (Morgan Kaufmann, San Francisco, 1995), pp. 108–114
15. S.P. Coy, R. Balasubramanian, B.L. Golden, O. Kwon, H. Beirjandi, Using neural networks to predict the degree of underpricing of an initial public offering, in *Proceedings of the Third International Conference on Artificial Intelligence Applications on Wall Street* (R.S. Freedman, ed.), Software Engineering Press, Gaithersburg, MD, pp. 223–231, 1995
16. D. Enke, S. Thawornwong, The use of data mining and neural networks for forecasting stock market returns. Expert Syst. Appl. **29**, 927–940 (2005)
17. A. Ghandar, Z. Michalewicz, M. Schmidt, T. Thuy-Duong, R. Zurbrugg, Computational intelligence for evolving trading rules. IEEE Trans. Evol. Comput. **13**(1), 71–86 (2009)
18. M. Grinblatt, C.Y. Hwang, Signaling and the pricing of new issues. J. Finance **44**, 393–420 (1989)
19. M. Hall, G. Holmes, E. Frank, Generating rule sets from model trees, in *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence: Advanced Topics in Artificial Intelligence* (Springer, London, 1999), pp. 1–12
20. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: An update. SIGKDD Explor. **11**(1), 10–18 (2009)
21. K.W. Hanley, The underpricing of initial public offerings and the partial adjustment phenomenon. J. Financ. Econ. **34**(2), 231–250 (1993)
22. R.S. Hansen, P. Torregrosa, Underwriter compensation and corporate monitoring. J. Finance **47**(4), 1537–1555 (1992)

23. B.A. Jain, O. Kini, On investment banker monitoring in the new issues market. J. Bank. Finance **23**, 49–84 (1999)
24. B.A. Jain, B.N. Nag, Artificial neural network models for pricing initial public offerings. Decis. Sci. **26**(3), 283–299 (1995)
25. B. Kirkulak, C. Davis, Underwriter reputation and underpricing: Evidence from the Japanese IPO market. Pac. Basin Finance J. **13**(4), 451–470 (2005)
26. J. Korczak, P. Lipinski, P. Roger, Evolution strategy in portfolio optimization, in *Artificial Evolution*, ed. by P. Collet, C. Fonlupt, J.-K. Hao, E. Lutton, M. Schoenauer. Lecture Notes in Computer Science, vol. 2310 (Springer, Berlin, 2002), pp. 156–167
27. H. Leland, D. Pyle, Informational asymmetries, financial structure and financial intermediation. J. Finance **32**, 371–387 (1977)
28. C.C. Lin, Y.T. Liu, Genetic algorithms for portfolio selection problems with minimum transaction lots. Eur. J. Oper. Res. **185**(1), 393–404 (2008)
29. A. Ljungqvist, W. Wilhelm, IPO pricing in the dot-com bubble. J. Finance **58**, 723–752 (2003)
30. T. Loughran, J.R. Ritter, Why has IPO underpricing changed over time? Financ. Manag. **33**, 5–37 (2004)
31. M. Lowry, M.S. Officer, W. Schwert, The variability of IPO initial returns. J. Finance **65**(2), 425–465 (2010)
32. C. Luque, D. Quintana, J.M. Valls, P. Isasi, Two-layered evolutionary forecasting for IPO underpricing, in *Proceedings of the Eleventh conference on Congress on Evolutionary Computation* (IEEE Press, Piscataway, 2009), pp. 2374–2378
33. W.L. Megginson, K.A. Weiss, Venture capitalist certification in initial public offerings. J. Finance **46**(3), 799–903 (1991)
34. F. Mendes, J. Duarte, A. Vieira, A. Gaspar-Cunha, Feature selection for bankruptcy prediction: a multi-objective optimization approach, in *Soft Computing in Industrial Applications*, ed. by X.-Z. Gao, A. Gaspar-Cunha, M. Köppen, G. Schaefer, J.Wang (Springer, Berlin, 2010), pp. 109–115
35. D. Meng, A neural network model to predict initial return of Chinese SMEs stock market initial public offerings, in *Proceedings of the IEEE International Conference on Networking, Sensing and Control (ICNSC)* (IEEE Press, Piscataway, 2008), pp. 394–398
36. R. Mitsdorffer, J. Diederich, C. Tan, Rule extraction from the technology IPOs in the US stock market, in *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP'02)* (IEEE Press, Piscataway, 2002), pp. 2328–2334
37. J. Moody, C.J. Darken, Fast learning in networks of locally tuned processing units. Neural Comput. **1**, 281–294 (1989)
38. R.J. Quinlan, Learning with continuous classes, in *5th Australian Joint Conference on Artificial Intelligence*, ed. by A. Adams, L. Sterling (World Scientific, Singapore, 1992), pp. 343–348
39. D. Quintana, C. Luque, P. Isasi, Evolutionary rule-based system for IPO underpricing prediction, in *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation* (ACM, New York, 2005), pp. 983–989
40. B. Reber, B. Berry, S. Toms, Predicting mispricing of initial public offerings. Intell. Syst. Account. Finance Manag. **13**, 41–59 (2005)
41. I. Rechenberg, Cybernetic solution path of an experimental problem, in *Royal Aircraft Establishment: Farnborough*, Hampshire, UK, Library Translation p. 1122 (1965)
42. J.R. Ritter, I. Welch, A review of IPO activity, pricing, and allocations. J. Finance **57**(4), 1795–1828 (2002)
43. S.J. Robertson, B.L. Golden, G.C. Runger, E.A. Wasil, Neural network models for initial public offerings. Neurocomputing **18**, 165–182 (1998)
44. D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors. Nature **323**, 533–536 (1986)
45. H.P. Schwefel, Kybernetische evolution als strategie der exprimentellen forschung in er strmungstechnik. Master's thesis, Technical University of Berlin (1965)
46. K. Shin, Y. Lee, A genetic algorithm application in bankruptcy prediction modeling. Expert Syst. Appl. **23**, 321–328 (2002)

47. S.B. Smart, C.J. Zutter, Control as a motivation for underpricing: A comparison of dual and single-class IPOs. J. Financ. Econ. **69**, 85–110 (2003)
48. A.J. Smola, B. Schölkopf, A tutorial on support vector regression. Stat. Comput. **14**, 199–222 (2004)
49. L.G. Tian, Financial Regulations, Investment Risks, and Determinants of Chinese IPO Underpricing. Working paper, Peking University Management School (2003)

Chapter 8
Bayesian Networks for Portfolio Analysis and Optimization

Simone Villa and Fabio Stella

Abstract Portfolio analysis studies the impact of economic and financial scenarios on the performance of an investment portfolio, while portfolio optimization concerns asset allocation to achieve a trade-off between risk and return. In this chapter we exploit the interplay between modern portfolio theory and Bayesian networks to describe a new framework for portfolio analysis and optimization. Bayesian networks provide an effective way to interface models to data, allow efficient evidential reasoning, while their graphical language offers an intuitive interface by which the analyst can elicit his/her knowledge. The proposed framework leverages on evidential reasoning to understand the behavior of an investment portfolio in different economic and financial scenarios. It allows to formulate and solve a portfolio optimization problem, while coherently taking into account the investor's market views. The Bayesian network framework for portfolio analysis and optimization is instantiated on the DJ Euro Stoxx 50 Index. Examples of portfolio analysis and optimization, exploiting evidential reasoning on Bayesian networks, are presented and discussed.

8.1 Introduction

Portfolio analysis and portfolio optimization are basic problems in computational finance. They have been intensively studied over the last 60 years, while several relevant contributions are available in the specialized literature. Portfolio

S. Villa
University of Milano – Bicocca, 20126 Milano, Italy

Saint George Capital Management, 6907 Lugano, Switzerland
e-mail: simone.villa@disco.unimib.it

F. Stella (✉)
University of Milano – Bicocca, 20126 Milano, Italy
e-mail: stella@disco.unimib.it

optimization originates from the seminal paper of Markowitz [15] who introduced the *mean–variance investment framework*. This conventional approach to portfolio optimization consists of two steps. The first one concerns distributional assumptions about the behavior of stock prices, while the second one is related to the selection of the optimal portfolio depending on some objective function and/or utility function defined according to the investor's goal. This conceptual model in the past proved to be useful even if many drawbacks have been pointed out by finance practitioners, private investors, and researchers. The basic formulation introduced by Markowitz has been extended in the specialized literature by taking into account additional moments of the portfolio's return distribution and by developing necessary conditions on the utility function of investors [9]. It is increasingly understood that the investor's experience, i.e., his/her qualitative and quantitative knowledge on economy, finance and financial markets, is a key factor for success. Indeed, the knowledge on: the likelihood of future events, the outlook on finance and economy, the coexistence of different asset pricing theories, and the security-driving forces, can be fruitfully exploited to formulate and solve the portfolio optimization problem. In such a context the Bayesian approach offers a set of powerful tools for implementing the main tasks of the *investment management process*: (1) specification of the investment objectives, (2) policy settlement, (3) specification of the portfolio strategy, (4) portfolio optimization, and (5) measurement and evaluation of the portfolio's performance (see [3]). Bayesian methods allow the investor to account for the uncertainty about the parameters of the return generating process, to incorporate prior beliefs in the decision–making process, and to provide a way to control and limit the sensitivity of the optimal portfolio allocation to the input parameters by shrinking the estimate of the market parameters toward the investor's prior. Therefore, Bayesian methods definitely address a deficiency of the standard statistical measures in conveying the economic significance of the information contained in a data sample (see [36]).

In this chapter we describe how Bayesian networks can be used to combine the investor's knowledge with the market data to perform portfolio analysis and optimization. Section 8.2 presents the basics of portfolio modeling and optimization. Bayesian networks are presented in Sect. 8.3 together with the framework for portfolio analysis and optimization. Section 8.4 describes a case study which concerns portfolio analysis and optimization on the DJ Euro Stoxx 50 financial market.

8.2 Portfolio Modeling and Optimization

This section describes the mean–variance portfolio framework together with the first five steps of *The Prayer* [24], a recipe used by finance practitioners to model and manage the Profit and Loss (P&L) distribution of theirs portfolios over a given investment horizon. The first five steps of the prayer recipe are:

1. Quest for invariance
2. Estimation of the invariants distribution
3. Projection of the invariants distribution into the future
4. Pricing
5. Aggregation

To better describe the steps of the prayer recipe we let the financial market to consist of m securities, $\mathbf{P}_t$ be an m-dimensional vector whose components are associated with the spot price at time t for the m securities, T be the time when the portfolio allocation decision has to be made, τ be the *investment horizon*, and $\tilde{\tau}$ be the *estimation interval*. The portfolio modeling task consists of forecasting the portfolio's P&L distribution at the end of the investment horizon $T+\tau$, see [16] for a detailed analysis.

8.2.1 Quest for Invariance

The forecasting model of the financial market behavior is developed through:

1. *Identification of the risk drivers*. A *risk driver* of a security is modeled with a random variable D sharing the following properties: (1) it fully specifies the price of the security at time t and (2) it follows a homogeneous stochastic process. Examples of risk drivers are: the log-price for stocks, the yield to maturity for bonds, and the log-price of the underlying security together with the logarithm of its implied volatility for equity options.
2. *Extraction of the invariants from the risk drivers*. An *invariant* I is a shock that steers the stochastic process associated with the risk driver D. It shares the following properties: (1) can be modeled by a set of independent and identically distributed random variables and (2) becomes known at time $t+\tilde{\tau}$. To connect the invariant I to the risk driver D, the following random walk model is used:

$$I_t = h(D_t) - h(D_{t-\tilde{\tau}}), \tag{8.1}$$

   where D_t and I_t are the value of the risk driver and the value of the invariant at time t, while h is an invertible deterministic function. More flexible processes, taking into account autocorrelations, stochastic volatility, and long memory, can be used. Examples of invariants are: compounded returns for equities, changes of the yield to maturity for bonds, compounded returns of the underlying security, and changes of the log-implied volatility for equity options (note that in this case (8.1) is associated with a multivariate random walk).

8.2.2 Estimation of the Invariants Distribution

This step concerns the estimation of the probability distribution of the J-dimensional *invariants vector* $\mathbf{I}$ which must not depend on any specific time value t. This step consists of:

1. *Estimation*. The invariants distribution for the next step $f_{\mathbf{I}_{T+\tilde{\tau}}}$ is approximated by fitting an empirical distribution to the invariants time series. This fitting problem depends on the sample size and quality (e.g., outliers and missing data) [16]. Nonparametric estimators perform well in the case where the number of observations is large, while when the number of observations is small, a parametric approach is preferred. In the case where very few observations are available shrinkage estimators are used. This step must properly account for the risk of using an estimate of the invariants distribution instead of the true one.
2. *Dimension reduction*. The number of invariants J can be very large and thus the estimation and management of the invariants distribution may become intractable. Therefore, the *linear factor model* dimension reduction technique is used to decompose the J-dimensional invariants vector $\mathbf{I}$ as follows:

$$\mathbf{I} = \mathbf{c} + \mathbf{B}\mathbf{F} + \mathbf{U}, \tag{8.2}$$

   where $\mathbf{c}$ is a J-dimensional *vector of constants*, $\mathbf{F}$ is an M-dimensional *factors vector* (with $M \ll J$), $\mathbf{B}$ is a $J \times M$ *matrix of factor loadings* which links the factors vector $\mathbf{F}$ to the invariants vector $\mathbf{I}$, while $\mathbf{U}$ is a J-dimensional *vector of residuals*. According to [23] the available linear factor modeling approaches are:

   - *Dominant-plus-residual model*. Where the term $\mathbf{c}+\mathbf{B}\mathbf{F}$ in (8.2) is optimized to explain the largest portion of the variability of $\mathbf{I}$ under a set of constraints on $\mathbf{c}$, $\mathbf{B}$, and $\mathbf{F}$, and according to a general measure of fitness (e.g., the generalized R-square). This class includes: pure models, time series, cross section, and statistical models.
   - *Systematic-plus-idiosyncratic model*. Where the term $\mathbf{c}+\mathbf{B}\mathbf{F}$ in (8.2) is selected such that (1) the residuals $\mathbf{U}$ are uncorrelated with each other and (2) the residuals $\mathbf{U}$ are uncorrelated with the factors $\mathbf{F}$.

8.2.3 Projection of the Invariants Distribution into the Future

The distribution of the risk drivers $\mathbf{D}_{T+\tau}$ at the end of the investment horizon $T+\tau$ is obtained by projecting the invariants distribution $f_{\mathbf{I}_{T+\tilde{\tau}}}$, recovered from the previous step, to the investment horizon $T+\tau$. The projection operation can be implemented in different ways. If the risk drivers evolve according to a random walk then the first two moments of the projected distribution can be obtained through recursion, while projecting the full distribution is more challenging, but can still be accomplished by the Fourier transform technique [1].

8.2.4 Pricing

The securities price vector $\mathbf{P}_{T+\tau}$ is computed with the following pricing formula:

$$\mathbf{P}_{T+\tau} = g(\mathbf{D}_{T+\tau}; i_T), \tag{8.3}$$

where g is the *pricing function*, i_T is the available information at time T, i.e. terms and conditions. The risk drivers together with terms and conditions completely determine the price of a security. If we are interested in obtaining the P&L distribution of the securities, we simply subtract the current prices vector $\mathbf{P}_T$ from the future prices vector $\mathbf{P}_{T+\tau}$. No closed form solution exists to compute the distribution of the prices under general conditions and thus it is customary to approximate the pricing formula (8.3) by Taylor expansion (see [42]). It is worthwhile to notice that (8.3) makes the assumption that a unique price of the security exists, while this is not always the case. The pricing step must take into account the liquidity risk and its implications.

8.2.5 Aggregation

The knowledge of the P&L distribution of the securities at the end of the investment horizon allows to compute the P&L distribution A of the portfolio $\mathbf{a}$ as follows:

$$A_{T+\tau} = \mathbf{a}'(\mathbf{P}_{T+\tau} - \mathbf{P}_T). \tag{8.4}$$

This computational step may be time consuming, while involving the solution of multiple integrals. A possible solution consists in describing the financial market with a set of scenarios and in applying scenario aggregation.

8.2.6 Mean–Variance Portfolio Optimization

This section describes the *mean–variance two-step portfolio optimization model* [16, 22]. This model requires the knowledge of the investor's profile, specified by: (1) the current portfolio and horizon, (2) the objective function $O_{\mathbf{a}}$, and (3) the attitude toward risk summarized by the index of satisfaction S. This index depends on the investor's objective function $O_{\mathbf{a}}$ and thus on the portfolio vector $\mathbf{a}$. Feasible portfolios $\mathbf{a}$, i.e., satisfying a set of constraints C, are compared according to the investor's index of satisfaction S, while the optimal portfolio $\mathbf{a}^*$ solves the following maximization problem:

$$\mathbf{a}^* = \underset{\mathbf{a}\in C}{\operatorname{argmax}} S(\mathbf{a}). \tag{8.5}$$

The *mean–variance portfolio optimization* framework focuses on the first two moments of the probability distribution of the investor's objective function and makes the assumption that $S(\mathbf{a}) \approx f(E[O_{\mathbf{a}}], \mathrm{Var}[O_{\mathbf{a}}])$ for some well–behaved function f. The portfolio $\mathbf{a}^*$, which solves the optimization problem (8.5), belongs to the following parametric family:

$$\mathbf{a}(v) = \underset{\substack{\mathbf{a}\in C, \\ \mathrm{Var}[O_{\mathbf{a}}]=v}}{\operatorname{argmax}} E[O_{\mathbf{a}}]. \tag{8.6}$$

The optimization problem (8.6) is the mean–variance two–step formulation of the portfolio optimization problem, while its solution set is called the *mean–variance efficient frontier* [15]. This formulation solves the problem (8.5) through the following steps: (1) computation of the mean–variance efficient frontier and (2) solution of the following one–dimensional constrained optimization problem:

$$\mathbf{a}^* = \mathbf{a}(\hat{v}) = \underset{v\geq 0}{\operatorname{argmax}} S(\mathbf{a}(v)). \tag{8.7}$$

The optimal portfolio $\mathbf{a}^*$ maximizes the expected value of the investor's objective function subject to the investor's constraints on the risk level v. It is worthwhile to notice that when the investor's objective function is the terminal wealth while the initial capital is nonnull, the portfolio optimization problem can be formulated in terms of the linear returns $\mathbf{L}$ and weights $\mathbf{w}$ vectors:

$$\mathbf{w}(v) = \underset{\substack{\mathbf{w}\in C, \\ \mathbf{w}'\mathrm{Cov}[\mathbf{L}]\mathbf{w}=v,}}{\operatorname{argmax}} \mathbf{w}'E[\mathbf{L}], \tag{8.8}$$

where $E[\mathbf{L}]$ and $\mathrm{Cov}[\mathbf{L}]$ are, respectively, the expected value and the covariance matrix of the vector of linear returns $\mathbf{L}$ at the end of the investment horizon. If a budget constraint is introduced ($\mathbf{w}'\mathbf{1} = 1$) and short-selling is not allowed ($\mathbf{w} \geq \mathbf{0}$), then the optimization problem is quadratic and thus can be solved analytically.

8.3 Bayesian Networks for Portfolio Analysis and Optimization

8.3.1 Basic Definitions and Notation

Bayesian networks (BNs) [13, 27, 31] are probabilistic models which allow the efficient description and management of joint probability distributions. BNs proved to be a useful tool for combining informal expert knowledge with statistical techniques for distribution evaluation. They are specifically designed for cases when the vector of random variables can have considerable dimension and/or it is difficult to come up with traditional parametric models of the joint probability distribution. Bayesian networks are popular in computer science, with specific reference to data mining and machine learning. Indeed, classical machine learning methods like hidden Markov models, neural networks, and Kalman filters can be considered as special cases of Bayesian networks [25]. An extension of Bayesian networks to model discrete time stochastic processes is offered by Dynamic Bayesian networks [26], while recently continuous time Bayesian networks [29] have been proposed to cope with continuous time stochastic processes. Bayesian networks and Bayesian network classifiers have been previously applied to finance [7, 10, 21, 28, 30, 41].

A Bayesian network (BN) over n variables, $\mathbf{X} = (X_1, \ldots, X_n)$, consists of a directed acyclic graph $\mathbf{G} = (\mathbf{V}, \mathscr{E})$ and a set of conditional probability distributions Θ. Each node $i \in \mathbf{V}$ is associated with a discrete random variable $X_i \in \mathbf{X}$ with $x_i \in \{1, \ldots, k_i\}$ possible outcomes. The directed links $\mathscr{E} \subseteq \mathbf{V} \times \mathbf{V}$ of $\mathbf{G}$ specify assumptions of conditional dependence and independence between random variables according to the *d-separation criterion* [31]. Each variable $X_i \in \mathbf{X}$ is associated with a conditional probability distribution, $P(X_i|pa(X_i)) \in \Theta$, where $pa(X_i)$ denotes the parents of node X_i, i.e., the set of variables which directly influence the random variable X_i. A BN encodes a joint probability distribution over the random vector $\mathbf{X}$ while the set of conditional probability distributions, Θ, allows the joint probability distribution over $\mathbf{X}$ to factorize as follows:

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|pa(X_i)). \tag{8.9}$$

A BN is often constructed by exploiting the cause–effect relations between entities of a given problem domain. This is done in such a way that nodes are associated with entities, while the links usually represent cause–effect relations (direct or indirect) between entities. However, many real–world problems involve continuous variables, and thus BNs have been extended to cope with continuous random variables by discretization, i.e., by partitioning the range of each random variable into a fixed set of intervals. This technique is adequate, but very often results in a considerable loss of accuracy or leads to very large conditional probability tables (CPTs). Another solution exploits standard families of probability density functions, fully specified by means of a finite and usually small number of parameters. It is worthwhile to mention that the Gaussian probability distribution is the most used due to its analytical tractability [39].

8.3.2 Evidential Reasoning

BNs allow efficient evidential reasoning in highly dimensional problem domains. In the case where X is the query variable, $\mathbf{E}$ is the vector of the observed variables, $\mathbf{e}$ is their observed value, and $\mathbf{Y}$ is the random vector of the unobserved variables, the query $P(X|\mathbf{E}=\mathbf{e})$ can be written as follows:

$$P(X|\mathbf{E}=\mathbf{e}) = \alpha P(X,\mathbf{E}=\mathbf{e}) = \alpha \sum_{i=1}^{|\mathbf{Y}|} P(X,\mathbf{E}=\mathbf{e},Y_i), \tag{8.10}$$

the sum over all possible where α is a normalization constant ensuring that $P(X|\mathbf{E}=\mathbf{e})$ adds up to 1, while the sum is over all possible assignments of the unobserved variables $\mathbf{Y}$. It is worthwhile to mention that according to (8.9) the terms $P(X,\mathbf{E}=\mathbf{e},Y_i)$ in (8.10) can be written as products of conditional probabilities. Therefore, a query $P(X|\mathbf{E}=\mathbf{e})$ can be answered with *inference by enumeration*, i.e., by computing sums of products of conditional probabilities according to the BN model.

However, even in the case where the random variables are binary, the joint probability distribution has size $O(2^n)$, and thus the time required for the summation over the joint probability distribution is exponential in the number of variables. The inference by enumeration algorithm can be significantly improved, through dynamic programming, by eliminating repeated calculations (see [6]). The full summation over discrete variables (or integration for continuous variables) is called *exact inference* and is known to be NP–hard [5]. Some efficient algorithms to perform exact inference exist in the case where the BN model is a *polytree*, i.e., its directed acyclic graph has at most one undirected path between any pair of nodes. Exact inference in polytrees is linear in the size of the network; while the most popular inference algorithm for polytrees is the Pearl's *message passing algorithm* [31]. This algorithm has been extended by Lauritzen and Spiegelhalter [14] to obtain the *junction tree algorithm*, which works on general BN models. Other exact inference algorithms are: cycle–cutset conditioning and variable elimination [6].

The intractability of exact inference for general BN models calls for approximate inference algorithms: direct sampling, rejection sampling, and likelihood weighting [39]. However, the most interesting approximate inference algorithm belongs to the class of Monte Carlo Markov Chains (MCMC) and exploits the Gibbs sampling and Metropolis–Hastings algorithms [11].

8.3.3 Learning Bayesian Networks

The directed acyclic graph, $\mathbf{G} = (\mathbf{V},\mathscr{E})$, and the set of conditional probability distributions, Θ, of the BN model can be learned from data through parametric and structural learning. *Parametric learning* is concerned with the estimation of the elements of the set Θ when the directed acyclic graph $\mathbf{G}$ is known, while *structural learning* is concerned with the selection of both components of the BN model, i.e., the directed acyclic graph $\mathbf{G}$ and the set of conditional probability distributions Θ. Parametric and structural learning can be developed for complete or missing data arising from partial observability of the random vector **X**, i.e., when some random variables cannot be directly observed (hidden variables) or have not been measured during the data collection process. BN learning is a complex task and its detailed treatment is out of the scope of this chapter.

Parametric learning when BN structure is known and all the variables are observed is briefly presented to better describe the Bayesian network framework for portfolio analysis and optimization. To this extent we let

$$DS = \{(x_{t,1}, \ldots, x_{t,n}), t = 1, \ldots, N\}, \tag{8.11}$$

be a data set consisting of N complete observations over n discrete variables $\mathbf{X} = (X_1, \ldots, X_n)$, each with $x_i \in \{1, \ldots, k_i\}$ possible outcomes. In such a setting the task of parametric learning consists of estimating the terms $P(X_i|pa(X_i))$ in (8.9). This task can be performed by maximum likelihood estimation (MLE) or Bayesian learning. In parametric learning it is customary to make the assumption that the conditional probabilities are fully parameterized, i.e., each term $P(X_i|pa(X_i)) = \theta_{X_i|pa(X_i)}$, $i = 1,...,n$, can be selected without any constraint on $pa(X_i)$. MLE maximizes the log-likelihood [12] of the data set DS (8.11):

$$\begin{aligned} LL(DS;\theta,\mathbf{G}) &= P(DS|\theta) = \sum_{t=1}^{N} \log P(x_{t,1}, \ldots, x_{t,n}|\theta) = \sum_{t=1}^{N}\sum_{i=1}^{n} \log \theta_{x_{t,i}}|pa(x_{t,i}) \\ &= \sum_{i=1}^{N} \sum_{x_i, pa(x_i)} \eta(x_i, pa(x_i)) \log \theta_{x_i|pa(x_i)}, \end{aligned} \tag{8.12}$$

where $\eta(X_i, pa(X_i))$ represents the number of observations such that X_i and its parents $pa(X_i)$ have a fixed assignment. The closed form solution of (8.12) is

$$\hat{\theta}_{x_i|pa(x_i)} = \frac{\eta(x_i, pa(x_i))}{\sum_{x_i=1}^{k_i} \eta(x_i, pa(x_i))}. \tag{8.13}$$

It is worthwhile to notice that the MLE approach suffers from the sparse data problem, while Bayesian learning offers a valid alternative.

8.3.4 The Bayesian Network Framework

Several models for portfolio optimization allow the investor to exploit qualitative market views as described in the specialized literature (see [19]). The first contribution is due to Black and Litterman [4] who made two significant contributions to the problem of portfolio optimization: (1) the Capital Asset Pricing Model [40] which is used to estimate the prior distribution of the asset returns and (2) a way to specify and blend the complete/partial investor's views with the prior distribution of the asset return. The process to input the investor's views described in [4] consists of the following steps: (1) identification of the eligible universe, market capitalization, and time series of returns for each asset class and for the risk–free asset, used to compute the covariance matrix of excess returns, (2) specification of the investor's views to be used to compute the estimates of the returns, and (3) portfolio optimization. The most interesting extensions of the Black and Litterman model are: Qian and Gorman [35], where the conditional-marginal factorization is used to input views on volatilities, correlations, and expectations; Almgren and Chriss [2], who provided a framework to rank views on expectations; Meucci [17], where the market views are used without any preprocessing; Pezier [32], where full/partial views on expectations and covariances are used; Meucci [18], where different confidence levels and multiple users are pooled to obtain a posterior consistent with the most general views; Meucci [19], who handles views and allows stress testing in derivative markets by nonlinearly mapping generic risk factors to the P&L distribution. Finally, Meucci [21] described a methodology to stress test a set of risk drivers under minimal information [18] and presented a novel consistency algorithm which extends the Bayesian network approach presented in [37].

In this chapter we go a step further and propose a Bayesian network framework for portfolio analysis and optimization which consists of the following layers:

1. *Bayesian network layer*. Links the key factors with the invariants, while providing a compact description of the portfolio as discussed in Sect. 8.2.
2. *Transformation layer*. The investor may be interested not merely in the P&L probability distribution at the end of the investment horizon, but also in the probability distribution of the risk drivers, prices or returns.
3. *Aggregation layer*. It models the behavior of a set of securities in different scenarios and allows to stress test a portfolio by combining a Bayesian network model with the investor's market views.

8.3.4.1 Bayesian Network Layer

This layer links the factors vector $\mathbf{F}$ to the invariants vector $\mathbf{I}$ by using a *factor analysis BN model* (Fig. 8.1) which factorizes the joint probability $P(\mathbf{F},\mathbf{I}) = P(\mathbf{F})P(\mathbf{I}|\mathbf{F})$. The directed acyclic graph $\mathbf{G} = (\mathbf{V},\mathscr{E})$ is bipartite. Indeed, $\mathbf{V} = (\mathbf{F},\mathbf{I}) = (F_1,\ldots,F_M,I_1,\ldots,I_J)$ consists of *factor nodes* and *invariant nodes*, while the set of arcs is defined as follows: $\mathscr{E} = \{(F_k,I_j)\colon 1 \leq k \leq M,\ 1 \leq j \leq J\}$.

The vectors $\mathbf{F}$ and $\mathbf{I}$ can be modeled with multivariate normal random variables, i.e., $P(\mathbf{F}) \sim \mathscr{N}(\mathbf{0},\mathbb{I})$ and $P(\mathbf{I}|\mathbf{F} = \mathbf{x}) \sim \mathscr{N}(\mathbf{a}+\mathbf{B}\mathbf{x},\Sigma)$, where $\mathbb{I}$ is the identity matrix, while Σ is a diagonal matrix. It is worthwhile to mention that when the number of factors is much smaller than the number of invariants, i.e., $M \ll J$, the BN model explains a highly dimensional vector $\mathbf{I}$ through a linear combination of low–dimensional features $\mathbf{F}$. It is customary to make the simplifying assumption of isotropic noise, i.e., to assume that $\Sigma = \alpha\mathbb{I}$ for some scalar α. Furthermore, it turns out that the maximum likelihood estimate (MLE) of the factor loading matrix $\mathbf{B}$ (8.2)

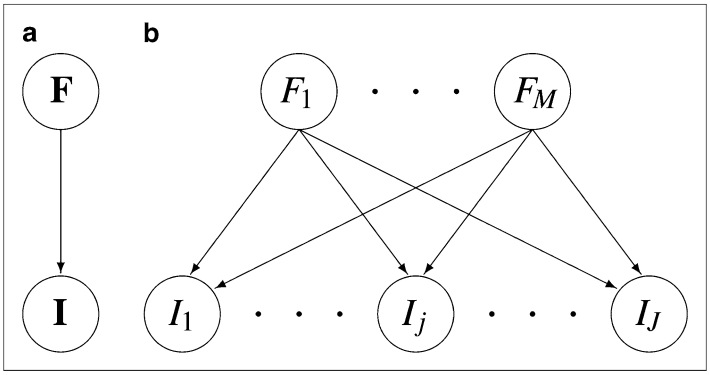


Fig. 8.1 Factor analysis Bayesian network

is given by the first M principal eigenvectors of the sample covariance matrix, with scalings determined by the eigenvalues and sigma. Note that the classical principal component analysis (PCA) is obtained by taking the limit $\alpha \to 0$ [38, 43]. Given the directed acyclic graph $\mathbf{G} = (\mathbf{V}, \mathscr{E})$ and by assuming that all random variables are observed, the parametric learning is performed by MLE. It is important to note that for Gaussian random variables, the MLE of the mean and covariance are, respectively, the sample mean and the sample covariance, while the MLE of the weight matrix is the least squares solution to the normal equations.

8.3.4.2 Transformation Layer

The BN model depicted in Fig. 8.1b allows to perform efficient evidential reasoning. Evidence concerning the nodes associated with the factors vector $\mathbf{F}$ can be introduced into the BN model, while exact inference algorithms can be used to compute the posterior probability over the invariants vector $\mathbf{I}$.

However, in practice, we are usually interested not only in analyzing the distribution of the invariants vector $\mathbf{I}$ but also rather in the distribution of other key variables of the securities, such as the risk drivers, the prices, or the returns. Therefore, the proposed Bayesian network framework for portfolio analysis and optimization is augmented with a second layer which links the J nodes associated with the invariants vector $\mathbf{I}$ to R nodes associated with the components of the *security key variable vector* $\mathbf{K}$. The components of the invariant vector $\mathbf{I}$ and the components of the securities key variables vector $\mathbf{K}$ can be fully or partially connected to each other. This layer is not constrained to be represented with a BN model, but rather it is a visual representation of one or more generic functions. When we are interested in modeling the risk drivers given the invariants, we can use the h^{-1} function, see (8.1), while when the prices have to be modeled we first determine the risk drivers and then apply the g function to them, see (8.3). The main advantage of using a generic function to link the invariants vector $\mathbf{I}$ to the securities key variables vector $\mathbf{K}$ is to speed up evidential reasoning on the Bayesian network.

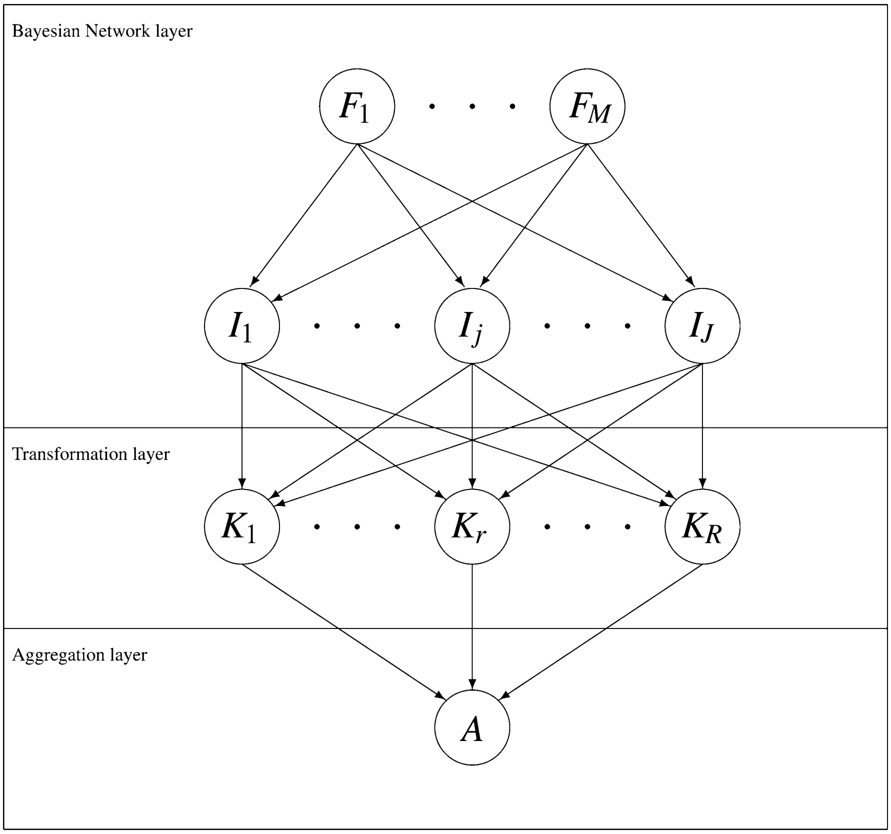


Fig. 8.2 Conceptual model of the Bayesian network framework for portfolio analysis and optimization. The *Bayesian Network layer* links the factors vector $\mathbf{F}$ to the invariants vector $\mathbf{I}$, the *Transformation layer* links the invariants vector $\mathbf{I}$ to the securities key variables vector $\mathbf{K}$, the *Aggregation layer* links the securities key variables vector $\mathbf{K}$ to the portfolio variable A

8.3.4.3 Aggregation Layer

This layer links the securities key variables vector $\mathbf{K}$ to the *portfolio variable* A and thus definitely allows the investor to perform portfolio analysis and optimization. When the securities key variables vector $\mathbf{K}$ is used to model the securities prices (or their P&L), the portfolio value A (or its P&L) can be computed by aggregation, as described in Sect. 8.2.5. This layer allows both continuous and discrete random variables, while their probability distributions can be either parametric or nonparametric. However, in many circumstances, the use of parametric distributions may be too restrictive. Therefore, the implementation of the proposed Bayesian network framework for portfolio analysis and optimization relies on nonparametric discrete distributions. The conceptual model of the Bayesian network framework for portfolio analysis and optimization is depicted in Fig. 8.2.

8.4 Case Study: The DJ Euro Stoxx 50 Index

In this section, the Bayesian network framework for portfolio analysis and optimization is instantiated to the case where the Eurozone Blue-chips forming the *DJ Euro Stoxx 50 Index* are considered.

Two Bayesian network models (Fig. 8.2) have been instantiated. The first one consists of 15 *estimation factors* ($M = 15$), while the second one consists of 16 *interpretation factors* ($M = 16$), i.e., 6 *interpretation factors* associated with the European country indexes and 10 *interpretation factors* associated with the European GICS sector indexes. In both BN models the M factors are linked to 50 *invariants* ($J = 50$) which in turn are linked to 50 *security key variables* ($R = 50$), i.e., *projected prices*, by means of the *transformation layer*. The *security key variables* are aggregated into the investment portfolio by the *aggregation layer*.

Five years of *last prices* data, spanning from December 30th, 2005, to December 30th, 2010, are used. The framework has been implemented by exploiting the MATLAB software environment and the Bayes Net Toolbox [25]. The case study considers the position of a European equity investor interested in analyzing, stress testing, and optimizing the *tomorrow's return distribution* of his/her portfolio.

8.4.1 From Invariants to Projected Prices

The first four steps of the prayer recipe are described in the case where daily stock data and 1 day horizon are considered.

1. *Quest for invariance*. The risk drivers vector of the stocks is the log-prices vector, i.e., $\mathbf{D}_t = \ln(\mathbf{P}_t)$, while the invariants vector $\mathbf{I}_t$ is the vector of daily compounded returns. The function h in (8.1) is the natural logarithm ln and thus we can write the following:

$$\mathbf{I}_t = \ln(\mathbf{P}_t) - \ln(\mathbf{P}_{t-\tilde{\tau}}). \tag{8.14}$$

2. *Estimation of the invariants distribution*. Monte Carlo simulation is used to estimate the distribution of the invariants $f_{\mathbf{I}_{T+\tilde{\tau}}}$. A normal copula is used to generate 100,000 joint scenarios using the technique described in [20]. The problem's dimension can be reduced with: (1) *Factor Analysis Using Random Matrix Theory*. It allows to model the invariants $\mathbf{I}_t$ with a sum of $R \ll J$ informative factors plus a residual term represented with a sum of $J - R$ noise factors (see [8, 33] and [34]). The R largest eigenvalues of the empirical covariance matrix of a panel of invariants for a market consisting of n securities over t time periods are used. However, the number of the underlying factors is unknown. Thus, a cutoff point is computed to separate the R eigenvalues associated with the underlying factors from the remaining $J - R$ eigenvalues associated with the noise components. It is customary to choose the value of R by visual inspection

of the scree plot or by using ad hoc cutoff points of the eigenvalues distribution. (2) *Factors on Demand*. This framework offers the possibility to assign the portfolio return to *Q attribution factors* $\mathbf{Z} = (Z_1, \ldots, Z_Q)$, i.e., random variables correlated with the portfolio return (e.g., the cross-sectional industry factors). This is possible because the generation of the distribution for the attribution factors $f_{\mathbf{Z}|\mathbf{I}}$ (i.e., the conditional distribution of the securities key variables vector $\mathbf{Z}$ given the invariants vector $\mathbf{I}$) is constrained to have first generated the scenarios for the portfolio return, which are fully driven by the distribution of the invariants $f_{\mathbf{I}}$. Using the identity $f_{\mathbf{I},\mathbf{Z}} = f_{\mathbf{I}} f_{\mathbf{Z}|\mathbf{I}}$, we can estimate $f_{\mathbf{Z}|\mathbf{I}}$ through three steps: (1) quest for invariance, (2) application of conditional estimation techniques, and (3) application of the projection step (see [20]). It is important to mention that the *Factor Analysis Using Random Matrix Theory* method has been used for the BN model consisting of 15 estimation factors, while the *Factors on Demand* approach has been used for the BN model consisting of 16 interpretation factors.

3. *Projection of the invariants into the future*. The distribution of the next step invariants vector $f_{\mathbf{I}_{T+\tilde{\tau}}}$ is the desired distribution at the investment horizon. The estimation interval $\tilde{\tau}$ is equal to the investment horizon τ, i.e., $\tilde{\tau} = \tau = 1$ day.
4. *Pricing*. The choice of the pricing function g in (8.3) allows to write the following pricing formula:

$$\mathbf{P}_{T+1} = \mathbf{P}_T e^{\mathbf{I}_{T+1}}. \tag{8.15}$$

8.4.2 Framework Instantiation

The modeling steps illustrated in Sect. 8.4.1 allow to obtain: the invariants forward–looking probability distribution, the key factors, forward–looking probability distribution, and the probability distribution of prices at the investment horizon. Therefore, the three layers of the Bayesian network framework for portfolio analysis and optimization can be instantiated.

1. *Bayesian network layer*. It is implemented through the BN model described in Sect. 8.3.4.1, i.e., each node of the BN is associated with a discrete random variable. The components of the factors vector $\mathbf{F}$ are discretized into the following states: *low*, *medium*, and *high*. The components of the invariants vector $\mathbf{I}$ are discretized to a greater granularity, i.e., 100 states. MLE parametric learning is used to estimate the parameters of the conditional probability distributions of $\mathbf{K}$ and $\mathbf{I}$.
2. *Transformation layer*. The securities key variables vector $\mathbf{K}$ is the vector of the securities prices $\mathbf{P}$. This layer allows to analyze how the investor's views on the key factors affect the distribution of the security's forward–looking price.
3. *Aggregation layer*. This layer is used to formulate and solve the portfolio optimization problem on the DJ Euro Stoxx 50 market. The optimal portfolio is ensured to reflect the investor's views on the key factors.

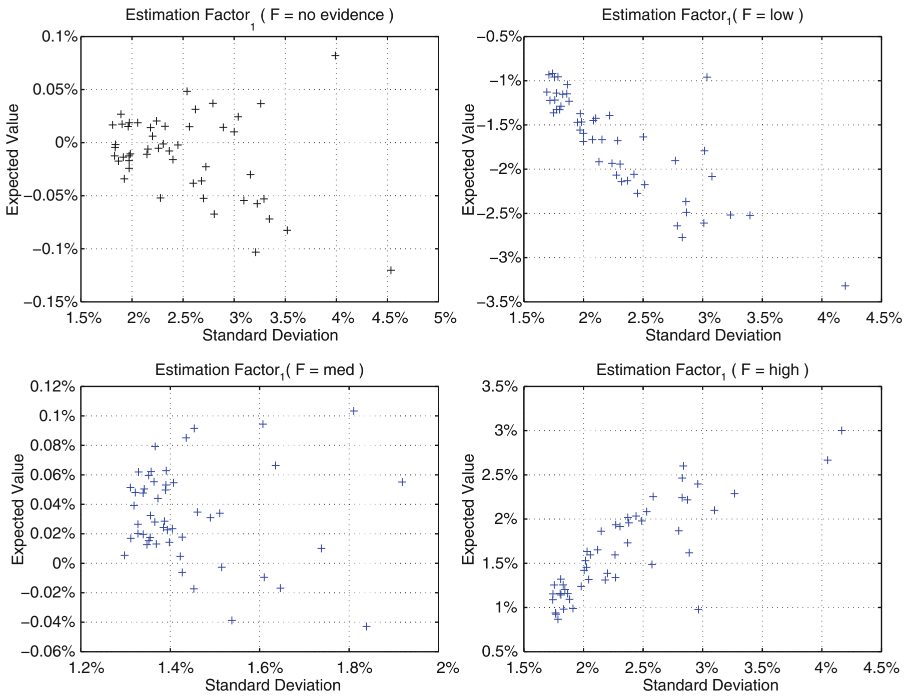


Fig. 8.3 Stress test on estimation factors. Estimates of the expected value and standard deviation for the prior distribution of the returns for the securities of the DJ Euro Stoxx 50 Index (*top left*). Estimates of the expected value and standard deviation for the posterior distribution of the returns for the securities of the DJ Euro Stoxx 50 Index where the first factor F_1 is respectively evidenced to the state *low* (*top right*), *medium* (*bottom left*), and *high* (*bottom right*)

8.4.3 Evidential Reasoning: Market's Views

The constituents of the DJ Euro Stoxx 50 Index can now be analyzed and stress tested. To show how this is performed, a top down approach is adopted where basic moments, i.e., expected value and standard deviation of the projected returns of the securities, are analyzed first. Then, the attention is shifted to the full distribution of the return for each security. Examples of evidential reasoning on market views are:

1. *Views on estimation factors*. The first two moments of the projected returns of each security at the investment horizon are plotted. The best practice starts the portfolio analysis and optimization tasks by computing the expected value and the standard deviation when no evidence is available (top left of Fig. 8.3). Then, the principal estimation factor F_1 is evidenced to the state *low*, then to the state *medium*, and to the state *high* (Fig. 8.3). Evidential reasoning consists of computing the posterior distribution of the projected returns. The first eigenvector

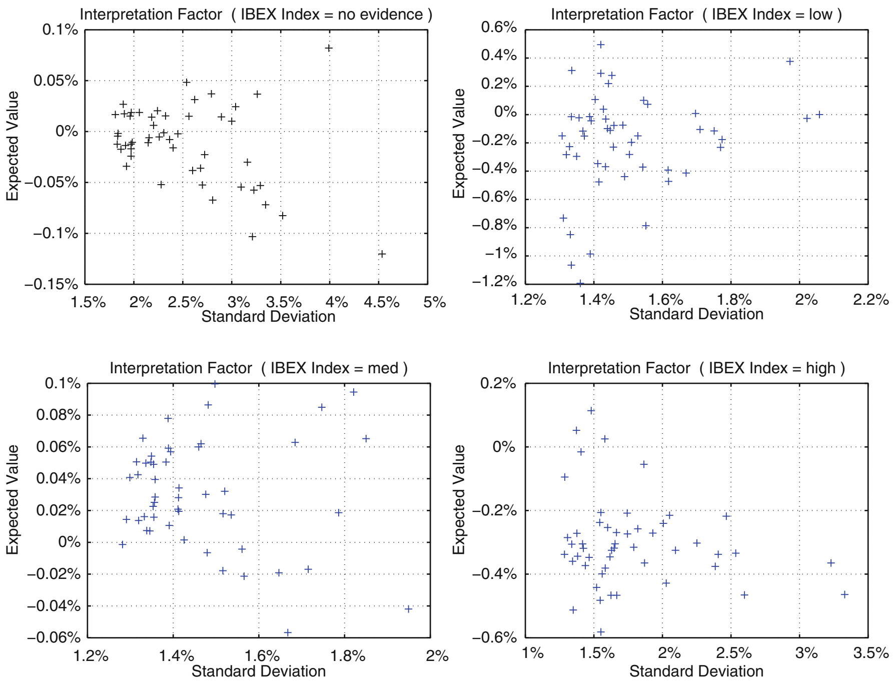


Fig. 8.4 Stress test on attribution factors. Estimates of the expected value and standard deviation for the distribution of the returns for the securities of the DJ Euro Stoxx 50 Index with no evidence (*top left*). Estimates of the expected value and standard deviation for the posterior distribution of the returns for the securities of the DJ Euro Stoxx 50 Index where the IBEX Index is respectively evidenced to the state *low* (*top right*), *medium* (*bottom left*), and *high* (*bottom right*)

F_1 is associated with a common factor driving the market. When F_1 is instantiated to the state *low*, the expected value of each security is negative; while when it is instantiated to the state *high*, the expected value of each security is positive. Furthermore, under both instantiations the returns of the securities are strongly correlated (Fig. 8.3).

2. *Views on interpretation factors*. Analysis and stress test of the reference market through the estimation factors can be difficult or simply not informative; moreover, analysis and stress test rely on the model and on the assumptions used to perform the dimension reduction step. Therefore, the estimation factors are replaced with two sets of *attribution factors* associated with the: (1) equity reference indexes of the leading European countries (i.e. CAC40 for France, DAX for Germany, FTSE MIB for Italy, IBEX for Spain, AEX for Netherlands, and BEL20 for Belgium) as shown in Fig. 8.4 and (2) ten European GICS sectors indexes as shown in Fig. 8.5. The comparison of Fig. 8.4 with Fig. 8.5 allows to conclude that stressing the Industrials GICS sector has a greater impact on the first two moments of the projected return distribution of the securities. This

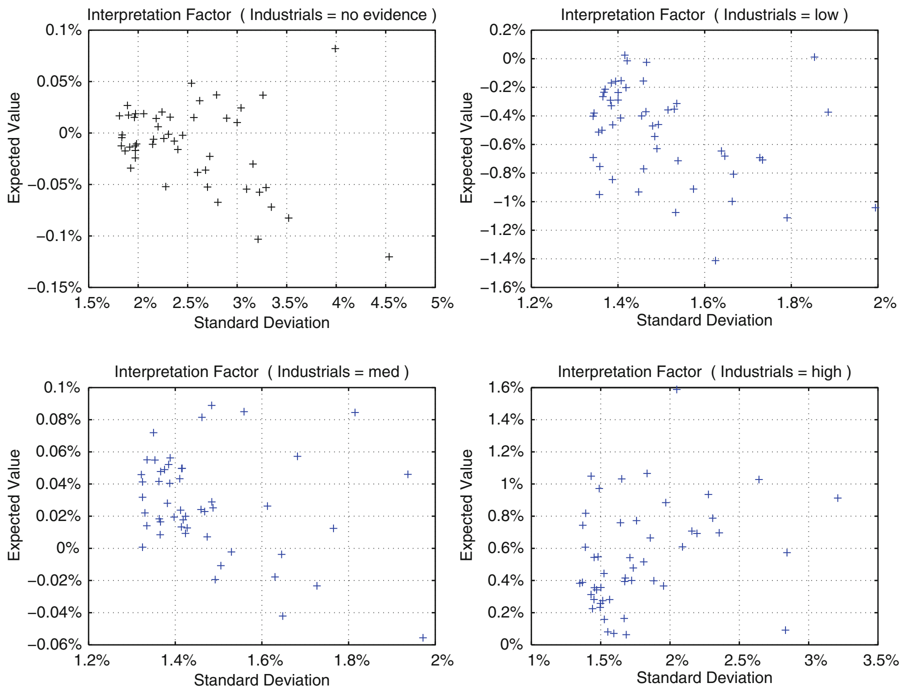


Fig. 8.5 Stress test on attribution factors. Estimates of the expected value and standard deviation for the distribution of the returns for the securities of the DJ Euro Stoxx 50 Index with no evidence (*top left*). Estimates of the expected value and standard deviation for the posterior distribution of the returns for the securities of the DJ Euro Stoxx 50 Index where the Industrials Index is respectively evidenced to the state *low* (*top right*), *medium* (*bottom left*), and *high* (*bottom right*)

effect is efficiently computed with the framework and can be used to analyze the behavior of each company under stress conditions. It is important to mention that the top-left graph of each figure does not change. This peculiarity of the framework is inherited from the *Factors on Demand* model: we can choose different sets of attribution factors without affecting the prior distributions.

3. *Single security*. The framework allows to analyze the distribution of the return of each security. Figure 8.6 shows the distribution of the returns for the first four securities, in alphabetical order, of the DJ Euro Stoxx 50 Index. Each graphic shows three distributions associated with: (1) no evidence, (2) evidence set to the state *low* for IBEX, and (3) evidence set to the state *low* for IBEX under the hypothesis that the distribution is normal. Figure 8.7 is similar to Fig. 8.6, but in this case it is the interpretation factor associated with the Industrials GICS sector to be evidenced to the state *low*. From the figures it is possible to conclude that the state *low* is more critical for the third security than the remaining ones.

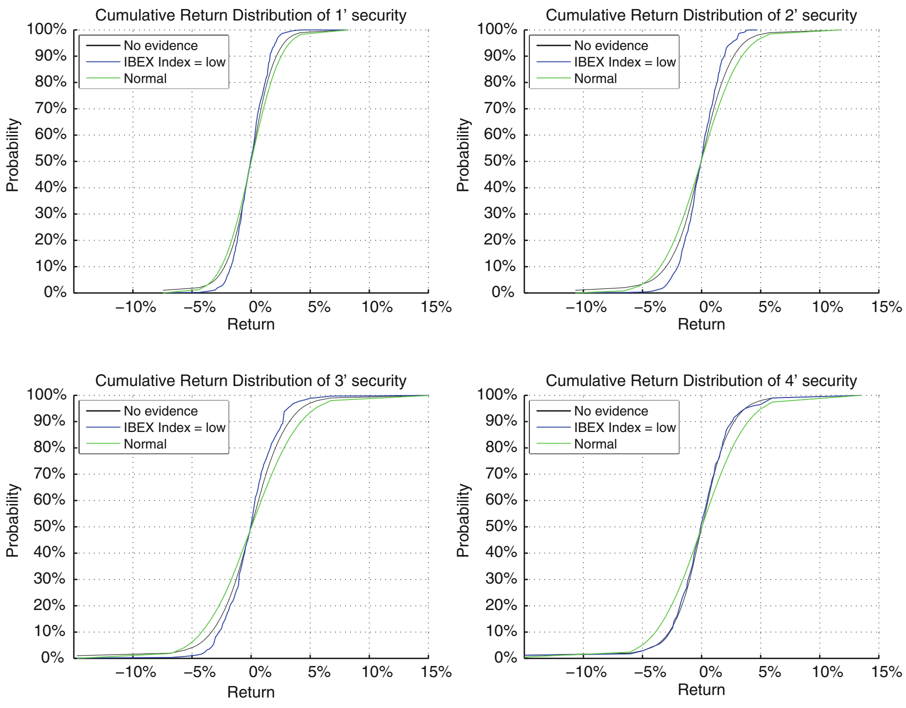


Fig. 8.6 Stress test on return distribution. Cumulative distribution of the returns for the first four securities of the DJ Euro Stoxx 50 Index where no evidence, normal distribution, and evidence on the IBEX Index is set to the state *low*

8.4.4 Evidential Reasoning: Portfolio Views

Stress testing the market to select the optimal portfolio under different scenarios is performed through evidential reasoning on portfolio views. Indeed, once the posterior distribution of the return of the securities has been recovered, it is possible to formulate and solve the portfolio optimization problem. To better clarify how the proposed framework allows to select the optimal portfolio, while encompassing the investor's market views, we adopt the mean–variance approach based on the projected linear returns according to (8.8). The efficient frontier together with the optimal portfolio composition when the IBEX Index is evidenced to the state *low* (*high*) are depicted on the left (right) hand side of Fig. 8.8. Figure 8.9 is similar to Fig. 8.8, but in this case the Industrials GICS sector is analyzed. The framework allows to construct portfolios with positive expected return under the analyzed scenarios.

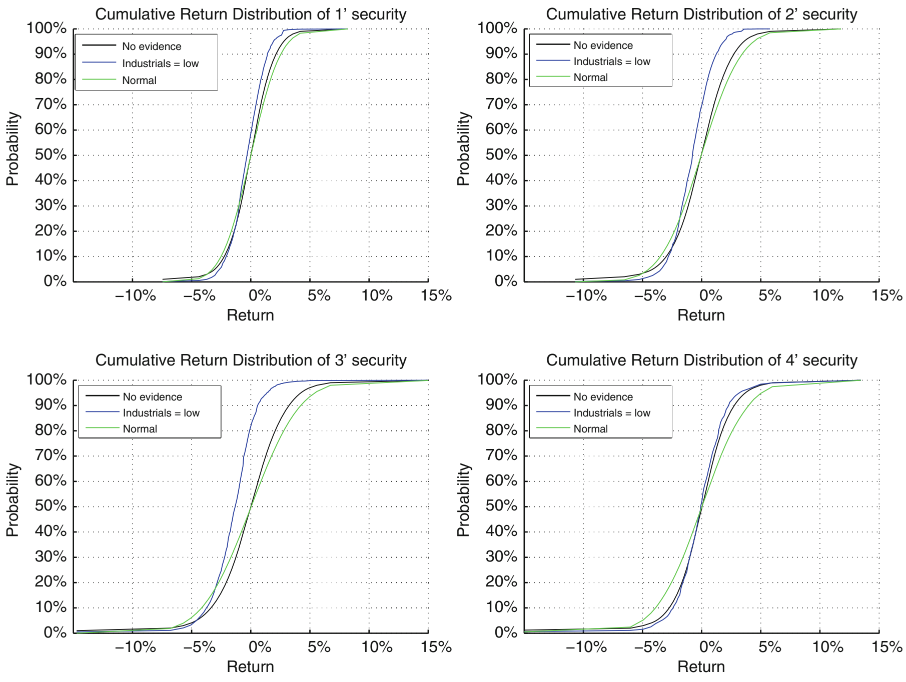


Fig. 8.7 Stress test on return distribution. Cumulative distribution of the returns for the first four securities of the DJ Euro Stoxx 50 Index where no evidence, normal distribution, and evidence on the Industrials Index is set to the state *low*

8.4.5 *Evidential Reasoning: Backtesting the Views*

A backtesting procedure is used to evaluate the impact of the views on the performance of the DJ Euro Stoxx 50 Index portfolio. A rolling window consisting of 4 years of daily data, spanning from January 4th, 2010, to December 30th, 2010, is used. The following assumptions are made: no transaction costs and market liquidity, i.e., it is always possible to trade at the *last price*. The backtesting procedure can be summarized as follows: (1) execution of the standard steps for portfolio modeling; (2) instantiation of the framework by using the *Countries* and *GICS sectors* attribution factors; (3) introduction of the qualitative views to the Bayesian network layer; (4) portfolio optimization; (5) computation of the linear returns for the optimal allocation.

The first analysis consists of comparing the daily linear returns of the DJ Euro Stoxx 50 (DJES 50) index with the returns of the selected portfolio allocation with a daily standard deviation equal to 0.5 and under the following scenarios for the IBEX stock market: no view, randomly selected view, wrong view, and correct view (Table 8.1). It is important to notice that in 72.62% of the days, the return achieved

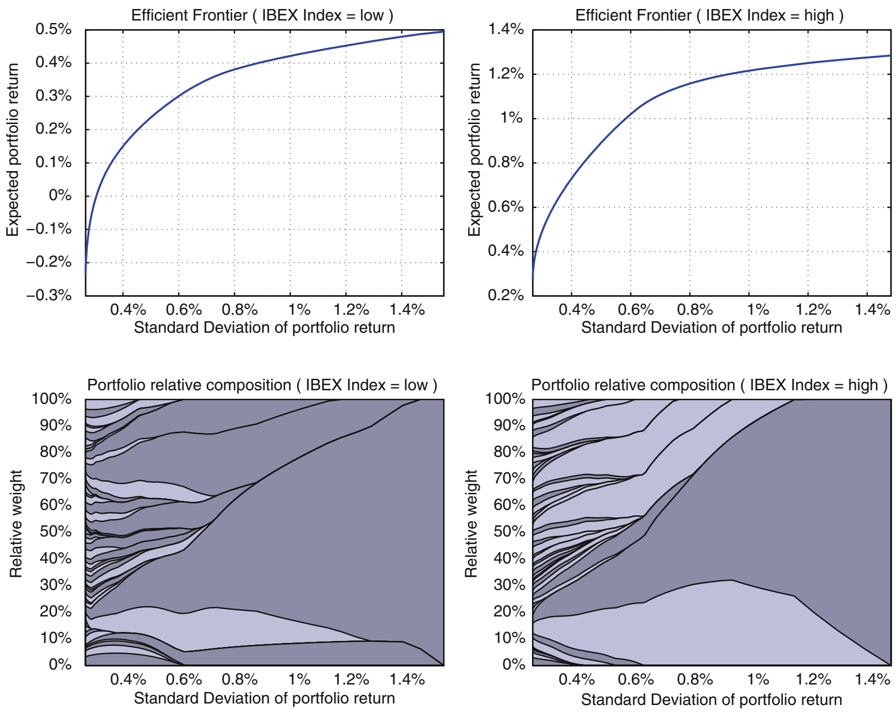


Fig. 8.8 Optimal portfolio allocation by using the market views. Efficient frontier and corresponding portfolio composition where the IBEX Index is evidenced to the state *low* (*left side*) and to the state *high* (*right side*)

by the portfolio selected under the correct view is greater than the return achieved by the DJES 50 Index. Table 8.2 concerns the same comparison when using views on the Industrials GICS sector.

The second analysis concerns the comparison of the entire distribution of the daily linear returns under the previous four scenarios. Tables 8.3 and 8.4 report on the main statistics of the distribution of returns for the portfolio allocation selected with the view on the IBEX stock market and on the Industrials GICS sector.

Tables 8.1–8.4 allow to conclude that the correct view is effective in selecting an optimal portfolio achieving a return which is greater than the one achieved without any market views. It is important to mention that the imputation of the correct view for each trading day is a complex task. However, the imputation is restricted to the state of a specific factor which can be well known by the investor. The empirical results confirmed that portfolio optimization with a qualitative view is a crucial part of the investment management process.

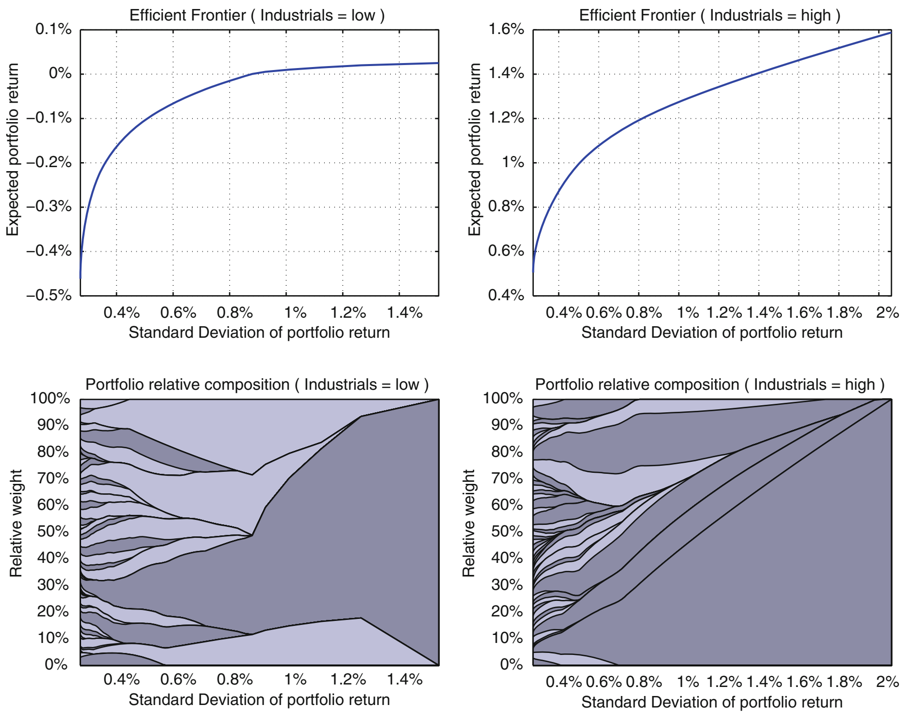


Fig. 8.9 Optimal portfolio allocation by using the market views. Efficient frontier and the corresponding portfolio composition where the Industrials Index is evidenced to the state *low* (*left side*) and to the state *high* (*right side*)

Table 8.1 Backtesting the views. Percentage of days where the return achieved by the portfolio associated with the row is greater than or equal to the return achieved by the portfolio associated with the column when the views on the IBEX Index are available

Returns ($\geq$)	DJES 50	No evidence	Wrong evidence
No evidence	57.14%		
Wrong evidence	28.17%	33.33%	
Correct evidence	72.62%	64.29%	71.83%

Table 8.2 Backtesting the views. Percentage of days where the return achieved by the portfolio associated with the row is greater than or equal to the return achieved by the portfolio associated with the column when the views on the Industrials GICS sector are available

Returns ($\geq$)	DJES 50	No evidence	Wrong evidence
No evidence	55.56%		
Wrong evidence	33.33%	28.17%	
Correct evidence	76.19%	64.68%	76.19%

Table 8.3 Backtesting the views. Summary statistics for the return distribution with views on the IBEX Index

Returns	DJES 50	No evidence	Wrong evidence	Correct evidence
Mean	−0.01%	0.06%	−0.29%	0.29%
St.dev.	1.50%	1.45%	1.56%	1.67%
Skewness	0.91	0.26	−0.54	2.40
Kurtosis	11.60	5.62	6.16	22.84
Minimum	−4.72%	−4.08%	−7.39%	−3.74%
Maximum	10.35%	7.51%	6.29%	14.49%
Median	−0.05%	0.03%	−0.17%	0.24%

Table 8.4 Backtesting the views. Summary statistics for the return distribution with views on the Industrials GICS sector

Returns	DJES 50	No evidence	Wrong evidence	Correct evidence
Mean	−0.01%	0.06%	−0.22%	0.26%
St.dev.	1.50%	1.41%	1.43%	1.48%
Skewness	0.91	0.28	−0.45	1.54
Kurtosis	11.60	5.62	5.46	13.07
Minimum	−4.72%	−4.08%	−5.03%	−4.00%
Maximum	10.35%	7.33%	6.47%	10.92%
Median	−0.05%	0.04%	−0.06%	0.14%

8.5 Conclusions

This chapter described portfolio analysis and optimization in the case where the investor is allowed to combine market data with his/her market views. The interplay between modern portfolio theory and Bayesian networks has been exploited to propose a new framework for portfolio analysis and optimization. The Bayesian network framework for portfolio analysis and optimization provides efficient ways to interface models to data and allows efficient evidential reasoning to understand the behavior of the investment portfolio in different economic and financial scenarios. We described how the investor can perform a what-if analysis on some financial factors to understand the behavior of the reference market under different stress testing conditions. Furthermore, the chapter described how the investor can formulate and solve the portfolio optimization problem to ensure that the selected portfolio allocation reflects the investor market views. The case study on DJ Euro Stoxx 50 Index emphasizes the relevance of evidential reasoning on estimation factors, interpretation factors, securities, and portfolios. The results confirm that the optimal portfolio, selected under the correct qualitative view, is effective. In conclusion, we can state that the proposed framework for portfolio analysis and optimization is a useful tool for those investors who need to integrate their quantitative and qualitative information (market view) with the available market data.

References

1. C. Albanese, K. Jackson, P. Wiberg, A new fourier transform algorithm for value at risk. Quant. Finance **4**, 328–338 (2004)
2. R. Almgren, N. Chriss, Optimal portfolios from ordering information. J. Risk **9**, 1–47 (2006)
3. D. Avramov, G. Zhou, Bayesian portfolio analysis. Annu. Rev. Financ. Econ. **2**, 25–47 (2010)
4. F. Black, R. Litterman, Asset allocation: Combining investors views with market equilibrium. Journal of Fixed Income, 7–18 (1991)
5. G. Cooper, The computational complexity of probabilistic inference using bayesian belief networks. Artif. Intell. **42**(2–3), 393–405 (1990)
6. R. Dechter, Bucket elimination: A unifying framework for reasoning. Artif. Intell. **113**, 41–85 (1999)
7. R. Demirer, R.R. Mau, C. Shenoy, Bayesian networks: A decision tool to improve portfolio risk analysis. J. Appl. Finance **16** (2006)
8. A. Edelman, Eigenvalues and Condition Numbers of Random Matrices. PhD thesis, Department of Mathematics, Massachussetts Institute of Technology (1989)
9. E.J. Elton, M.J. Gruber, S.J. Brown, *Modern Portfolio Theory and Investment Analysis* (Wiley, New York, 2009)
10. N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers. Mach. Learn. **29**, 131–163 (1997)
11. W.R. Gilks, S. Richardson, D.J. Spiegelhalter, *Markov Chain Monte Carlo in Practice* (Chapman and Hall, London, 1996)
12. D. Heckerman, D. Geiger, M. Chickering, Learning bayesian networks: The combination of knowledge and statistical data. Mach. Learn. **20**, 197–243 (1995)
13. F.V. Jensen, T.D. Nielsen, *Bayesian Networks and Decision Graphs* (Springer, Berlin, 2007)
14. S.L. Lauritzen, D.J. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems (with discussion). J. Roy. Stat. Soc. **50**, 157–224 (1988)
15. H. Markowitz, Portfolio selection. J. Finance **7**, 77–91 (1952)
16. A. Meucci, *Risk and Asset Allocation* (Springer, Berlin, 2005)
17. A. Meucci, Beyond black-litterman in practice: A five-step recipe to input views on non-normal markets. Risk **19**, 114–119 (2006)
18. A. Meucci, Fully flexible views: Theory and practice. Risk **21**, 97–102 (2008)
19. A. Meucci, Enhancing the black-litterman and related approaches: Views and stress-test on risk factors. J. Asset Manag. **10**, 89–96 (2009)
20. A. Meucci, Factors on demand. Risk **23**, 84–89 (2010)
21. A. Meucci, Fully flexible bayesian networks. http://ssrn.com/abstract=1721302 (2010)
22. A. Meucci, Linear vs. compounded returns-common pitfalls in portfolio management. GARP Risk Prof. "The Quant Classroom" series, 49–51 (2010)
23. A. Meucci, Review of linear factor models: Surprising common principles, the systematic-plus-idiosyncratic myth, and the misread relationship with financial theory, July (2010). http://ssrn.com/abstract=1635495
24. A. Meucci, The prayer - Ten-step checklist for advanced risk and portfolio management. GARP Risk Prof. "The Quant Classroom" series, 54–60/34–41 (2011)
25. K.P. Murphy, The bayes net toolbox for matlab. Comput. Sci. Stat. **33** (2001)
26. K.P. Murphy, Dynamic Bayesian Networks: Representation, Inference and Learning. PhD thesis, UC Berkeley, Computer Science Division (2002)
27. R.E. Neapolitan, *Learning Bayesian Networks* (Prentice Hall, NJ, 2003)
28. M. Neil, N. Fenton, M. Tailor, Using bayesian networks to model expected and unexpected operational losses. Risk Anal. J. **25**(4), 963–972 (2005)
29. U. Nodelman, C. Shelton, D. Koller, Continuous time Bayesian networks, in *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI)* (2002), pp. 378–387. http://robotics.stanford.edu/~nodelman/publications.html

30. T. Pavlenko, O. Chernyak, Credit risk modeling using bayesian networks. Int. J. Intell. Syst. **25**(4), 326–344 (2010)
31. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann, CA, 1988)
32. J. Pezier, Global portfolio optimization revisited: A least discrimination alternantive to black-litterman. ICMA Centre Discussion Papers in Finance (2007). http://www.icmacentre.ac.uk/files/pdf/dps/dp2007_07.pdf
33. V. Plerou, V. Gopikrishnan, B. Rosenau, L. Amaral, T. Guhr, E. Stanley, Random matrix approach to cross-correlations in financial data. Phys. Rev. E **65**, 1–17 (2002)
34. J.P.B.M. Potters, L. Laloux, Financial applications of random matrix theory: Old laces and new pieces. Acta Phys. Pol. B **36**(9), 2767–2784 (2005)
35. E. Qian, S. Gorman, Conditional distribution in portfolio theory. Financ. Analyst J. **57**, 44–51 (2001)
36. S.T. Rachev, J.S.J. Hsu, B.S. Bagasheva, F.J. Fabozzi, *Bayesian Methods in Finance* (Wiley, New York, 2008)
37. R. Rebonato, *Coherent Stress Testing: A Bayesian Approach to the Analysis of Financial Stress*, Wiley (2010). ISBN: 0470666013
38. S. Roweis, Z. Ghahramani, A unifying review of linear gaussian models. Neural Comput. **11**, 305–345 (1999). http://www.cs.nyu.edu/~roweis/papers/NC110201.pdf
39. S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd edn. (Prentice Hall, NJ, 2009)
40. W.F. Sharpe, Capital asset prices: A theory of market equilibrium under conditions of risk. J. Finance **3**, 425–442 (1964)
41. C. Shenoy, P.P. Shenoy, Bayesian Networks: A Decision Tool to Improve Portfolio Risk Analysis. Working paper, School of Business, University of Kansas (1998)
42. D. Stefanica, *A Primer for the Mathematics of Financial Engineering* (FE Press, New York, 2008)
43. M. Tipping, C. Bishop, Mixtures of probabilistic principal component analyzers. Neural Comput. **11**(2), 443–482 (1999)

Chapter 9
Markov Chains in Modelling of the Russian Financial Market

Grigory A. Bautin and Valery A. Kalyagin

Abstract We use Markov chains models for the analysis of Russian stock market. First problem studied in the chapter is concerned with multiperiod portfolio optimization. We show that known approaches applied for the Russian stock market produce the phenomena of nonstability and propose a new method in order to smooth it. The second problem concerns the structural changes in the Russian stock market after the financial crisis of 2008. We propose a hidden Markov chain model to analyze structural changes and apply it to the Russian stock market.

9.1 Markov Chain Models for the Russian Market

Portfolio optimization involves the allocation of funds between assets having different characteristics—profitability, risk, liquidity, etc. The main problem is that higher returns entail higher risk, and the investor operates under conditions of uncertainty. Thus, the investor faces the problem of finding the optimal strategy based on the preferences regarding return and risk ratio. In general, this problem is very complex and not amenable to rigorous mathematical description, and therefore different models are used to simplify the reality and allow formulating a set of specific recommendations.

A Markov chain is a sequence of random variables $x_1, x_2, x_3, \ldots$ that can take values from a countable set S and have the Markov property. The property can be formally defined in the following way:

$$P(X_{n+1} = x | X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = P(X_{n+1} = x | X_n = x_n),$$

G.A. Bautin (✉) • V.A. Kalyagin
National Research University Higher School of Economics, Lab LATNA, Russia
e-mail: gbautin@hse.ru; vkalyagin@hse.ru

i.e., conditional distribution of future states of the Markov chain depends only on the current state and is independent of all previous states. The set S of possible values of random variables X_n is called the state space. Another important characteristic of Markov chains is the transition matrix, which is defined as $p_{ij}^{(n)} = P(X_{n+1} = j|X_n = i)$. In what follows we use only homogenous Markov chain models where $p_{ij}^{(n)}$ are independent of n. There are also many different extensions of the basic Markov model, such as continuous time models or higher-order chains that permit dependencies between the future and more than one of the previous states of the chain. However, further we consider only the simplest model with discrete time and finite state space.

Markov models are widely used for analysis of the financial market (see [2, 6]). The returns of the assets in a market depend on many factors, some of which are invisible and not measurable. Generally, the asset returns can be treated as random variables. A model can assume that the asset returns are modulated by states of a Markov chain. There is now a considerable amount of publications on the topic, for instance, Stettner [8] provides an example on risk-sensitive portfolio optimization with completely or partially observed states; Bäuerle and Rieder [1] deal with a financial market with one bond and one stock where the expected return of the stock is modulated by an external finite state Markov chain; Çakmak and Özekici [3] provide an explicit solution for multiperiod portfolio optimization in a Markov-modulated market.

9.1.1 Modelling the Russian Financial Market

The Russian stock market is significantly different from developed Western and Asian markets. In recent years, as a result of many new companies entering the market, as well as the emergence of market access for a broad segment of the population (via Internet), some differences are smoothed out. However, the Russian market still has a number of characteristics.

In particular, most of the operations are speculative and long-term investments are the exception. There are also a lack of financial guarantee insurance, imperfection of the tax system, no liquid market for private debt assets, and a large number of investment institutions, imposing their services. These features suggest the use of new models or adjusting parameters and algorithms used in existing models. High volatility and unpredictability lead to an idea of using Markov chains to model the switching of market regimes and trying to guess the possible unobservable state of the market (that is a combination of factors influencing the returns). The chapter is organized as follows. First we study the multiperiod portfolio optimization problem for the Russian stock market. For our numerical experiments we use the stochastic market model suggested in [3]. As a result, we observe a phenomena of nonstability of the optimal multiperiod portfolio for the Russian market. We show that this nonstability is related with the choice of the states of the associated Markov chain model. To fix the instability phenomena we suggest to use a more sophisticated choice of the states of the model for Russian stock market. Second we deal with a structural change on the stock market after the financial crisis of 2008. To detect the structural changes we use an appropriate hidden Markov chain model and show how it works for the Russian stock market.

9.1.2 Multiperiod Portfolio Optimization Using a Markov Chain Model

We start with the model described in [3]. This model considers a market consisting of one risk-free asset, the yield of which is known in advance, and several risky assets which have random returns.

At any time, the market may be in one of the states belonging to the set of states E. Let the state of the market at time n (or the period n) be Y_n. We assume that $Y = \{Y_n; n = 1, 2, \dots, \}$ is a Markov chain with state space E and the transition matrix Q.

Return on the risk-free asset is known in advance for each of the states of the market, and equals $r_f(i)$, while returns of risky assets are random variables $R(i) = (R_1(i), R_2(i), \dots, R_m(i))$, where i is the state of the market.

Let $r_k(i) = E[R_k(i)]$ denote the mean return on asset k, and $\sigma_{kl}(i) = Cov(R_k(i), R_l(i))$ denote the covariance between yields of the assets k and l for the state of the market i. The excess (relative to the riskless asset) return of the asset k is defined as $R_k^e(i) = R_k(i) - r_f(i)$. It follows:

$$r_k^e(i) = E[R_k^e(i)] = r_k(i) - r_f(i) \tag{9.1}$$

$$\sigma_{kl}(i) = \mathrm{Cov}(R_k^e(i), R_l^e(i)) = \mathrm{Cov}(R_k(i), R_l(i)) \tag{9.2}$$

We introduce the following auxiliary vectors: $r(i) = (r_1(i), r_2(i), \dots, r_m(i))$ and $r^e(i) = (r_1^e(i), r_2^e(i), \dots, r_m^e(i))$. In addition, we assume that $E_i[Z] = E[Z|Y_0 = i]$ and $\mathrm{Var}_i(Z) = E_i[Z^2] - E_i[Z]^2$ are the mean and variance of Z, respectively, provided that the initial state of the market is i. For any matrix M, we assume that M' is a transposed matrix M.

Let X_n denote the capital available at time n, and $u = (u_1, u_2, \dots, u_m)$ is a column vector of investments in assets $(1, 2, \dots, m)$. Then the following dynamic equation may be considered:

$$X_{n+1} = r_f(Y_n)X_n + R^e(Y_n)'u. \tag{9.3}$$

The model assumes that we can borrow and lend at the rate of return of the riskless asset with no limits, and that the short selling is allowed for any asset in any period. It also does not take into account transaction costs, considering them to be negligible.

The following two equivalent dynamic programming problems can be formulated:

$$\begin{cases} E[X_T] \to \max \\ \quad \mathrm{Var}_i \leq \sigma \\ \quad X_{n+1} = r_f(Y_n)X_n + R^e(Y_n)'u \end{cases} \tag{9.4}$$

$$\begin{cases} \quad \mathrm{Var}_i \to \min \\ E[X_T] \geq \mu \\ \quad X_{n+1} = r_f(Y_n)X_n + R^e(Y_n)'u \end{cases} \tag{9.5}$$

These problems are nonseparable in the sense of dynamic programming. However, there is a parametric equivalent to these two problems:

$$\begin{cases} E[X_T] - \omega \mathrm{Var}_i(X_T) \to \max \\ X_{n+1} = r_f(Y_n)X_n + R^e(Y_n)'u \end{cases} \tag{9.6}$$

defined for $\omega > 0$. This problem is also nonseparable, but it can be solved with the help of an auxiliary problem (see [3]):

$$\begin{cases} E[-\omega X_T^2 + \lambda X_T] \to \max \\ X_{n+1} = r_f(Y_n)X_n + R^e(Y_n)'u \end{cases} \tag{9.7}$$

defined for $\omega > 0$ and all λ. The latter problem is separable in the sense of dynamic programming, and an explicit solution can be found for it. It turns out that any solution for it, where $\lambda = 1 + 2\omega E_i[X_T]$, is also the solution of problems (9.4) and (9.5).

Thus, the optimal investment strategy and the efficient mean–variance frontier can be obtained by solving problem (9.7). The following are the conclusions obtained by solving this problem, details and proofs are omitted and can be found in [3]. The optimal investment strategy is given by

$$u_n(i,x) = \left[\left(\frac{1 + 2\omega a_1(i)x_0}{2\omega(1 - 2b(i))} \right) \frac{\overline{Q}_g^{T-n-1}(i)}{\overline{Q}_f^{T-n-1}(i)} - r_f(i)x \right] V^{-1}(i)r^e(i) \tag{9.8}$$

for all $n = 0, 1, \ldots, T-1$, where for any matrix M and vector f, M_f is a matrix whose elements equal $M_f(i,j) = M(i,j)f(j)$. Furthermore, $\overline{M_f}$ is the vector obtained by summing the columns of the matrix M_f, i.e. $\overline{M_f}(i) = \sum_j M_f(i,j)$. In addition, we introduce the following auxiliary variables:

$$V(i) = E[R^e(i)R^e(i)'] = \sigma(i) + r^e(i)r^e(i)' \tag{9.9}$$

$$h(i) = r^e(i)'V^{-1}(i)r^e(i) \tag{9.10}$$

$$g(i) = r_f(i)[1 - h(i)] \tag{9.11}$$

$$f(i) = r_f(i)^2[1 - h(i)] \tag{9.12}$$

$$a_1(i) = \overline{Q_g}^{T-1}(i)g(i) \tag{9.13}$$

$$a_2(i) = \overline{Q_f}^{T-1}(i)f(i) \tag{9.14}$$

$$b(i) = \frac{1}{2}\sum_{k=1}^{T} Q^{k-1}\left(\frac{\left(\overline{Q_g}^{T-k}\right)^2}{\left(\overline{Q_f}^{T-k}\right)} \bullet h\right)(i), \tag{9.15}$$

where for all vectors a, b, and c, $((a/b)\bullet c)$ denotes the vector where $((a/b)\bullet c)(i) = (a(i)/b(i))c(i)$. In order to determine the optimal strategy in terms of σ and μ, the following formulas for ω should be used:

$$\omega = \sqrt{\frac{b(i)}{2\left[(1-2b(i))\sigma - [1 - 2b(i)a_2(i) - a_1(i)^2]x_0^2\right]}} \tag{9.16}$$

$$\omega = \frac{b(i)}{(1-2b(i))\mu - a_1(i)x_0}. \tag{9.17}$$

And the efficient frontier for the period T is defined by the following equation:

$$\mathrm{Var}_i(X_T) = \left(a_2(i) - \frac{a_1(i)^2}{(1-2b(i))}\right)x_o^2 + \frac{[(1-2b(i))E_i[X_T] - a_1(i)x_0]^2}{2b(i)(1-2b(i))}. \tag{9.18}$$

9.1.3 Analysis of the Model and Its Application to the Russian Market

The basic assumptions of the described above model are the following:

- There is a risk-free asset, and the investor can freely lend and borrow at the risk-free rate.
- Short-selling is allowed for all assets in all periods.
- The capital of the investor cannot be increased or decreased inside the planning horizon.
- Transaction costs are negligible.

The assumption of the existence of a riskless asset is not obvious for the Russian market. In fact, in this model, unlike many others, the risk-free asset and the risk-free rate are not just abstract theoretical concepts. This asset must actually exist, since the investor is assumed to be able to invest into it. Unlike the USA, where the government bonds are traditionally considered risk-free, there are virtually no such securities in the Russian market. Requirements for the riskless asset are as follows: (a) the returns of the asset should be predictable, ideally it must have a zero deviation, (b) the asset must be absolutely liquid, i.e., at any time you can buy or sell it in any volume, (c) the asset should be infinitely divisible. In our further numerical experiments, we suppose that risk-free asset exists and its return is zero for any state of the market. We consider daily data, so the time period is too short to get any positive return for any asset, which theoretically could be called a risk-free: government or blue chips bonds, bank deposits. Actually, in case of using any of these assets, the transaction costs would make the real return negative. In other words, we suppose that the investor has an unlimited financial leverage. In practice, for the most of optimal portfolios obtained with the model, the need for borrowed funds is quite low and well within the bounds provided by most broker leverage.

The possibility of short selling is offered by many brokerage companies. In the case of a short sale, the investor borrows the needed asset from the broker and sells it. Thus, at the end of the operation, the investor has to return not money, but the borrowed asset. There is always a risk that the investor will be able to return the borrowed assets, for example if the price of the asset has risen, and the investor does not have enough funds. In order to protect themselves from such losses, the brokerage firms establish certain requirements, for example they require investors to have a special account with a certain amount of money to cover the possible loss. Generally, the short selling is possible in a wide range of cases, but it leads to additional transaction costs.

The assumption that the capital is not changed inside the planning horizon seems to be quite plausible. Moreover, this assumption can be easily bypassed with a slight model modification. However, it is worth noting that in some cases the investor may decide to reduce the amount of capital invested in the asset, if things do not go well, or increase it otherwise. In addition, the model does not consider transaction costs, but in practice these costs can significantly influence the performance of the portfolio.

One of the weaknesses of the model is that it has a rather high sensitivity of the resulting expressions for the optimal strategy and the efficient frontier to the accuracy of calculations. Thus, minor change of one of the intermediate variables may, under certain conditions, lead to a significant shift in the efficient frontier. Another disadvantage of the model, which is typical for a large class of multiperiod models, is that the "excess" (i.e., unplanned) returns are automatically eliminated in the further periods. That is, if at some point the actual rate of return is higher than planned, the model is working to eliminate this excess return in the next iteration. From a mathematical point of view, the model works properly, but in practice in such a situation it is better to increase the planned yield. The model can be even dynamically updated with the recent data.

There is also one aspect that significantly influences the performance of the model—the method of determining the state of the market. In general, the method of determining the state must meet the following requirements:

- The criterion should be based on historical data, so we should be able to determine the state of the market at the moment: the investor cannot know what exactly will happen in the future.

- Since this model uses Markov chain, it is desirable that market state is determined by exactly one period of observation. Using data from several periods will distort the logic of the model.
- It is advisable that the returns of the assets in different states were significantly different.

The question of efficient state classification is discussed in the next sections.

9.1.4 Multiperiod Optimization on the Russian Stock Market

At the first step we use the method from [3] for determining the states of the market. Following this method the state in a given period depends on the number of assets whose prices have risen in the previous period. Thus, if all assets have fallen, it is considered that the market is in state s_1, if exactly one asset has increased in price—the market is in state s_2, etc. We want to apply the model to Russian financial market and need to choose several assets. In our numerical experiments we select assets from different industries and we apply the model to the shares of five Russian blue chips traded in MICEX:

- GAZP, LKOH—Oil and gas industry
- GMKN—Metallurgy
- SBER—Financial sector
- RTKM—Telecommunications

We choose 1 day as the length of the period to avoid intraday fluctuations on one hand, and to have enough data on the other. To avoid the influence of the financial crisis, we consider the time period from the beginning of 2009 until the end of 2011—totally 745 observations. Graphs of the value of assets, normalized by the first day, are shown in Fig. 9.1. The top two graphs correspond to Norilsk Nickel (GMKN) and Gazprom (GAZP), two graphs in the middle are Lukoil (LKOH) and Sberbank (SBER), and the bottom graph corresponds to Rostelecom (RTKM). We can see that the prices of certain assets fairly strongly correlated.

Let the initial wealth of the investor be $x_0 = 1$. We are going to apply the model with the investment horizon $T = 5$ and the target terminal return $\mu = 1.2763$ (that is an equivalent for a 5% daily return). Having five assets, we get six possible market states: from zero up to five asset prices can rise in a given period. Analyzing the historical data, we get the following transition matrix:

$$Q = \begin{pmatrix} 0.2746 & 0.1268 & 0.1479 & 0.1549 & 0.1549 & 0.1408 \\ 0.2432 & 0.1261 & 0.1712 & 0.1802 & 0.0991 & 0.1802 \\ 0.2072 & 0.2162 & 0.1171 & 0.1441 & 0.1351 & 0.1802 \\ 0.2015 & 0.1866 & 0.0970 & 0.2015 & 0.1716 & 0.1418 \\ 0.1140 & 0.1316 & 0.2105 & 0.2281 & 0.1228 & 0.1930 \\ 0.1092 & 0.1261 & 0.1765 & 0.1849 & 0.2437 & 0.1597 \end{pmatrix}.$$

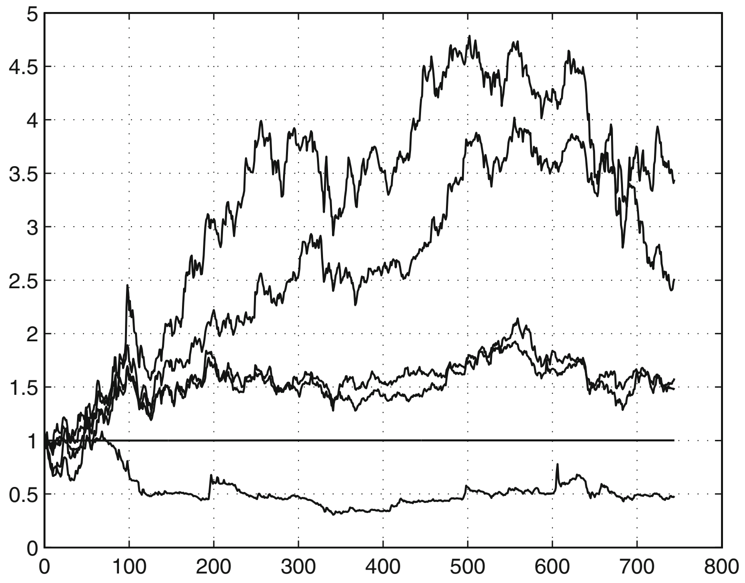


Fig. 9.1 Graphs of the value of assets, normalized by the first day. The *top* two graphs correspond to Norilsk Nickel (GMKN) and Gazprom (GAZP), two graphs in the *middle* are Lukoil (LKOH) and Sberbank (SBER), and the *bottom* graph corresponds to Rostelecom (RTKM)

Table 9.1 Expected returns of the assets

State	RTKM	GAZP	LKOH	SBER	GMKN
1	0.9811	0.9775	0.9811	0.9740	0.9765
2	0.9974	0.9881	0.9893	0.9879	0.9909
3	1.0002	0.9992	0.9986	1.0006	1.0004
4	1.0004	1.0066	1.0038	1.0065	1.0067
5	0.9996	1.0157	1.0147	1.0212	1.0164
6	1.0218	1.0209	1.0205	1.0273	1.0229

We can see that the transitions have no a straightforward interpretation related to the chosen states of the Markov model. The expected returns of the assets, depending on the market state, can be found in Table 9.1. We can see that the higher the number of the market state, the higher is the expected return of most of the assets. Now, we can use formulas (9.8) and (9.18) to calculate the optimal strategy and the efficient frontier (which is shown in Fig. 9.2).

The optimal strategy is shown in Table 9.2; the asset returns correspond to the period from 12/15/2011 until 12/21/2011. We can see that in the first period the investments are rather aggressive: the model proposes a portfolio with large short sellings. In the second period, there are no short sellings, but the total cost of the portfolio is higher than the capital that investor has at that moment, i.e., the investor has to borrow money to form this portfolio. The third period portfolio has large short

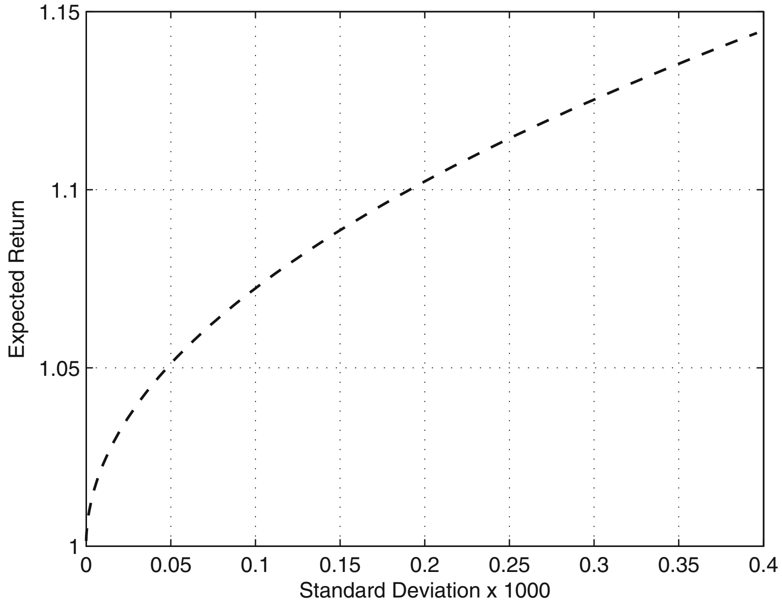


Fig. 9.2 The efficient frontier for dynamically optimized portfolio of five assets: GAZP, LKOH, GMKN, SBER, and RTKM. The number of periods $T = 5$

Table 9.2 The optimal strategy and the actual returns of the assets

Asset	u_1	r_1	u_2	r_2	u_3	r_3	u_4	r_4	u_5	r_5
RTKM	−1.547	0.981	2.323	1.038	−0.758	1.008	0.103	0.979	0.012	1.004
GAZP	−7.168	1.010	1.566	1.007	−3.512	0.971	0.682	1.027	0.008	1.006
LKOH	−2.656	0.993	0.778	1.011	−1.301	0.997	0.259	1.007	0.004	1.008
SBER	−2.277	0.987	1.199	1.010	−1.115	0.988	0.475	1.004	0.006	1.012
GMKN	−0.162	0.991	2.492	1.007	−0.079	0.971	0.266	0.998	0.013	1.021

sellings again, and the last two portfolios are more or less balanced. This strategy is unlikely to be practically applicable, but we should remember that we have chosen a rather high target return—about 28% in 5 days. Such an experiment is a good crash test for the model. The result of the applied strategy is shown in Fig. 9.3; the strategy suggested by the model leads to almost exactly the target return. Unfortunately this is not always the case. To evaluate the performance of the method we run the calculation 100 times with a different time period. The results are presented in Fig. 9.4. The dashed lines indicate the target portfolio return μ and theoretical standard deviation. The solid line is the actual portfolio return over 100 model runs on sequential time periods. It is clear from the figure that the performance of the proposed method is very low. The method applied for the Russian market manifests a high nonstability in the final return of the multiperiod portfolio. We will see in the next section that the result of multiperiod portfolio optimization for the Russian

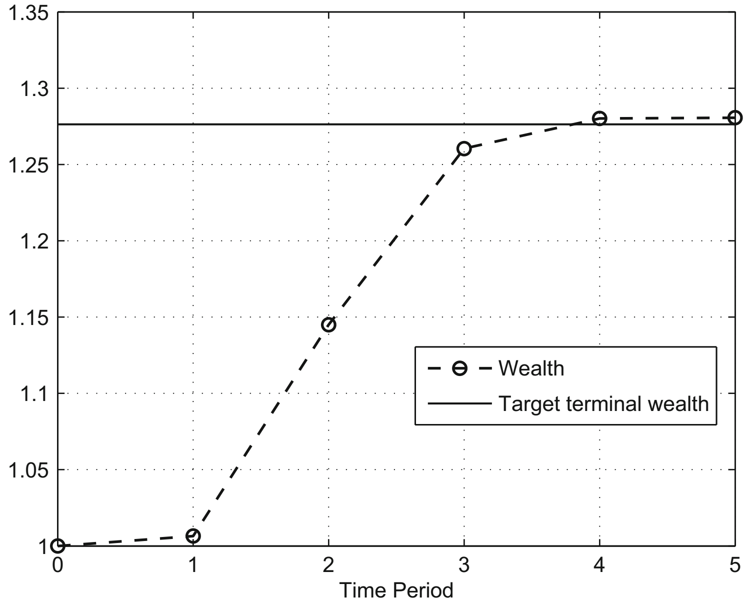


Fig. 9.3 The wealth of the investor over five periods. The strategy leads to almost exactly the planned return

stock market essentially depends on the choice of the states for the Markov chain model. An appropriate choice of the states makes the optimal portfolio more realistic and stable.

9.1.5 Efficient Market State Clustering for the Russian Market

To determine the most effective method of state clustering, we consider the optimization of a portfolio of three assets: Norilsk Nickel (GMKN), Lukoil (LKOH), and Rostelecom (RTKM). For a direct numerical test, we use the following technique. Having a particular method of state clustering, we determine the market state for each period of the historic data. Then, the model parameters are evaluated for each state: the asset return expectations, covariance matrix, transition matrix, that is, the market is analyzed in view of the selected states. The model is then applied to optimize a portfolio on the data that follows immediately after the historic data. To illustrate the results a graph can be built to indicate the portfolio returns over the entire planning horizon (i.e., the model performance) for each period of some time interval. Using this graph, one can visually evaluate how well the model works in different conditions, compare the planned and actual return μ, and see how the deviation of the actual yield is related to the planned risk σ (which, in turn, is determined from the efficient frontier).

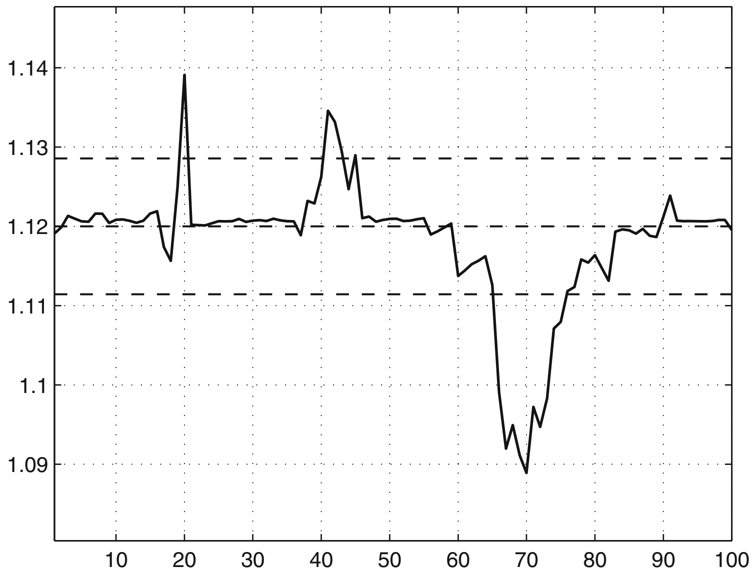


Fig. 9.4 The illustration of model performance with original method of state determination. The *dashed lines* indicate the target portfolio return μ and theoretical standard deviation. The *solid line* is the actual portfolio return over 100 model runs on sequential time periods

In addition to visual observations, the quality of the model can be measured by such values as the evaluation of the standard deviation of return on investment for the entire period (based on actual data), the average yield, and the yield spread. We use the following notation: μ is the expected return, σ is the standard deviation of the terminal wealth, μ_r is actual average yield, σ_r is the evaluation of the actual standard deviation, and D_r is the spreading of the actual yield (i.e., the difference between the highest and the lowest value).

We consider several methods of state determination. Let us call the approach described in [3] that takes into account the number of assets that rise in a given period the *basic method*. One of the drawbacks of this method is that the number of states equals the number of assets plus one, so if we want to build a portfolio of big number of assets, we have to have a really large historical data for proper model parameter estimation. Even if there is enough historical data, some global events such as financial crisis can affect the parameter estimation, because the nature of the market changes, and the parameter estimations can become unsuitable.

The model is applied 100 times, the historical data for parameter estimation always begins at 11.01.2009, and the end of the time frame slides from 03.08.2011 up to 21.12.2011, and the strategy is applied to the immediate next T observations. We consider different values of $T = 4, 8, 12, 16$ and $\mu = 1.06, 1.12, 1.19, 1.24$. The model performance for the *basic method* of state determination is shown in

Table 9.3 The model performance for the *basic method*

T	μ	σ	μ_r	σ_r	D_r
4	1.06	2.2376×10^{-4}	1.0571	2.3575×10^{-4}	0.1152
	1.12	8.9503×10^{-4}	1.1142	9.4299×10^{-4}	0.2304
	1.19	0.0022	1.1808	0.0024	0.3648
	1.24	0.0036	1.2284	0.0038	0.4608
8	1.06	1.8152×10^{-5}	1.0591	1.7143×10^{-5}	0.0251
	1.12	7.2609×10^{-5}	1.1181	6.8571×10^{-5}	0.0502
	1.19	1.8203×10^{-4}	1.1870	1.7190×10^{-4}	0.0795
	1.24	2.9044×10^{-4}	1.2363	2.7428×10^{-4}	0.1004
12	1.06	2.9607×10^{-6}	1.0596	1.7046×10^{-6}	0.0082
	1.12	1.1843×10^{-5}	1.1191	6.8184×10^{-6}	0.0164
	1.19	2.9689×10^{-5}	1.1886	1.7093×10^{-5}	0.0260
	1.24	4.7371×10^{-5}	1.2382	2.7274×10^{-5}	0.0328
16	1.06	3.4133×10^{-7}	1.0599	1.0125×10^{-7}	0.0018
	1.12	1.3653×10^{-6}	1.1198	4.0499×10^{-7}	0.0036
	1.19	3.4228×10^{-6}	1.1896	1.0153×10^{-6}	0.0057
	1.24	5.4613×10^{-6}	1.2395	1.6199×10^{-6}	0.0072

Table 9.3. The result is quite expected: the higher the target return, the higher the variation, and the bigger the planning period, the more accurate strategy the model is able to suggest.

The second method of state determination we are going to consider is the *average values method*. The state is determined by the average return of all assets, in the simplest case there are two states—the first state is when the average return of the assets is higher or equals 1, and the second state is when it is when the average return is less than 1. Obviously, we can set up any number of states by discretizing the average return. The proposed method is very simple: we calculate the maximum and minimum values of average returns, divide it by the number of states, and then for each new observation determine into which interval the average value falls.

For quality evaluation, we use the same technique as for the previous method, but now we calculate only the actual standard deviation of the terminal wealth σ_r for some of the T and μ values from the previous experiment and for different numbers of states. It turns out that when we have ten or more states, there is not enough data to estimate all the needed parameters, and some intermediate matrices become singular.

The model performance using the *average values method* is shown in Table 9.4. We can see that the accuracy of the model rises with the rise of number of states; however there is an obvious exception when the number of states equals 3. Most probably, in this case most of the observations fall into the "central state" 2, so the transition matrix and parameter estimations become improper for correct model functioning. Taking into consideration the calculation problems with high number of states, we can choose "the best" number of states $N_{\mathrm{st}} = 7$, and use it in further method comparison.

Table 9.4 Standard deviations of the terminal wealth σ_r for the *average values method* and various number of states N_{st}

T	μ	$N_{\text{st}} = 2$	$N_{\text{st}} = 3$	$N_{\text{st}} = 4$	$N_{\text{st}} = 5$	$N_{\text{st}} = 7$	$N_{\text{st}} = 9$
4	1.06	1.16×10^{-4}	0.0053	7.31×10^{-5}	5.87×10^{-4}	2.23×10^{-4}	6.56×10^{-6}
8	1.12	2.09×10^{-5}	0.0196	2.44×10^{-6}	2.92×10^{-4}	3.81×10^{-6}	5.79×10^{-9}
12	1.19	4.04×10^{-6}	0.0053	1.41×10^{-7}	2.13×10^{-4}	5.92×10^{-7}	3.55×10^{-12}
16	1.24	3.93×10^{-7}	0.1204	1.81×10^{-9}	8.25×10^{-5}	8.12×10^{-9}	2.80×10^{-15}

Table 9.5 Standard deviations of the terminal wealth σ_r for the *k-means method* and various number of clusters N_{cl}

T	μ	$N_{\text{cl}} = 2$	$N_{\text{cl}} = 3$	$N_{\text{cl}} = 4$	$N_{\text{cl}} = 5$	$N_{\text{cl}} = 7$	$N_{\text{cl}} = 9$
4	1.06	1.39×10^{-4}	7.23×10^{-5}	7.47×10^{-4}	4.78×10^{-4}	7.87×10^{-6}	6.87×10^{-6}
8	1.12	5.55×10^{-5}	8.46×10^{-6}	3.93×10^{-4}	3.01×10^{-6}	3.92×10^{-7}	2.12×10^{-8}
12	1.19	8.40×10^{-6}	5.95×10^{-7}	3.52×10^{-4}	3.44×10^{-6}	9.76×10^{-11}	4.68×10^{-11}
16	1.24	1.40×10^{-6}	4.82×10^{-8}	1.62×10^{-4}	1.94×10^{-7}	3.11×10^{-11}	4.15×10^{-13}

The third method of state determination that we consider is the *k-means method*. We apply the k-means clustering to vectors of asset returns, so the historical data is divided into N_{cl} clusters. After that, every new observation can be attributed to one of the clusters. This is done by calculating the Euclidean distance between the new observation vector and the centers of the clusters, and finding the closest cluster. It should be noted that in case of using cluster analysis the partitioning should be performed several times, and the most efficient partition should be chosen, since the initial cluster centers are set up randomly, affecting the final result.

The performance of the model with states determined using the *k-means method* is shown in Table 9.5. Again, the accuracy of the model grows with the growth of number of states, but there is a computational limit for efficient model parameters estimation. We can also notice that with $N_{\text{cl}} = 4$ the performance is worse than with $N_{\text{cl}} = 3$ or $N_{\text{cl}} = 5$. This can be explained by the nature of the transition matrix that we get in case of $N_{\text{cl}} = 4$:

$$Q = \begin{pmatrix} 0.6833 & 0.0542 & 0.0846 & 0.1779 \\ 0.4130 & 0.2391 & 0.1522 & 0.1957 \\ 0.5185 & 0.0864 & 0.2963 & 0.0988 \\ 0.6515 & 0.0152 & 0.0833 & 0.2500 \end{pmatrix}.$$

The first state is obviously the most frequent state of the market: there are high probabilities of transitions into it and low probabilities of going out. With different number of clusters, the states are distributed more uniformly, and the model works better. Of course, this property cannot be extended to the general case, and is caused only by the nature of the data used for our experiments.

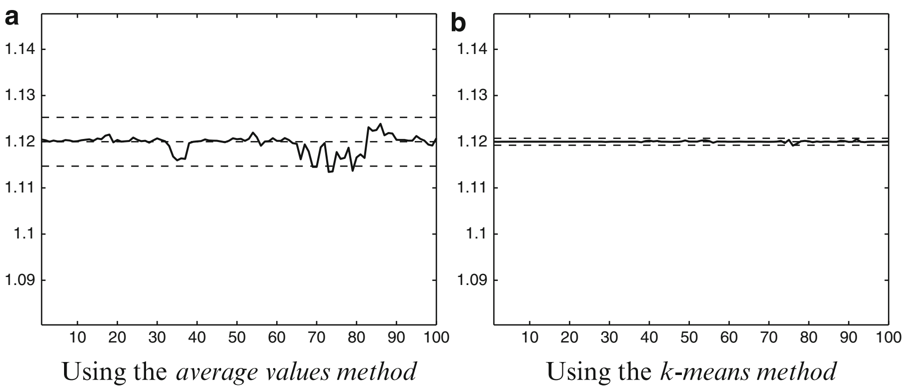


Fig. 9.5 The illustration of model performance with different state determination methods in use. The *dashed lines* indicate the target portfolio return μ and theoretical standard deviation. The *solid line* is the actual portfolio return over 100 model runs on sequential time periods

Although in some particular cases the *k-means method* shows a lower performance than the *average values method*, generally it looks more robust. The performance of all three discussed methods is shown in Fig. 9.5. The graphs are built for 100 model runs on sequential time periods with $\mu = 1.12$, $T = 8$, $N_{\text{st}} = 7$, and $N_{\text{cl}} = 7$; the graphs are in the same scale. The dashed lines indicate the target portfolio return μ and theoretical standard deviation. The solid line is the actual portfolio return over 100 model runs on sequential time periods. The figures clearly show that the *k-means method* manifests a good stability. It is preferable for state determination for Russian stock market, as it provides stable performance with a relatively small standard deviation.

9.2 Hidden Markov Chain Model for the Russian Market

The financial crisis of 2008 has had a big impact on the economics of Russia. It is a common opinion that the Russian stock market is not the same after the crisis as it was before. But it is not easy to measure this difference. The behavior of the indexes MICEX and RTSI of the Russian stock market before and after the crisis of 2008 is presented in Fig. 9.6. The period of observations is taken from 02.09.2003 to 14.12.2011 and the crisis period is fixed from 01.07.2008 to 05.12.2008.

As everywhere the crisis period is marked by a strongly correlated shut done of the indexes. From the first view the market behaviors before and after the crisis are similar. To analyze a structural change of the Russian stock market after the crisis of 2008 we use a hidden Markov chain model.

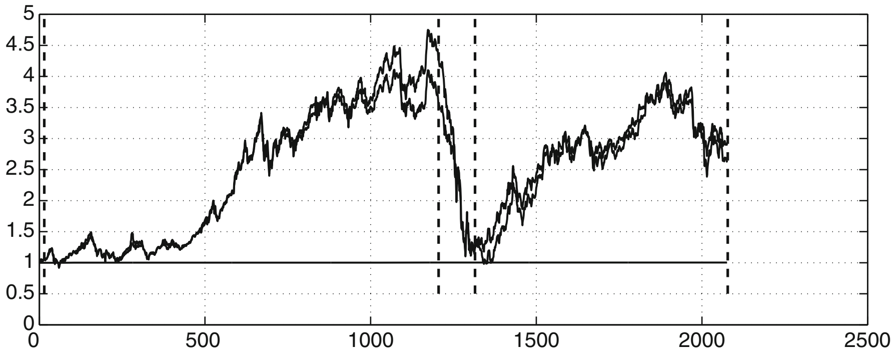


Fig. 9.6 The graph of MICEX and RTSI indexes

9.2.1 Hidden Markov Chain Model

A hidden Markov model is built of two random processes. The first one is an unobservable process, we cannot register it, but it can be described by the second random process, which gives us a set of signals—the observed sequence. In the simplest case, the observed process is discrete with a finite set of possible values $V = \{V_1, V_2, \dots V_M\}$, where M is the number of possible values. The hidden random process is a Markov process $q_1, q_2, q_3, \dots$ with N possible states from a state space $S = \{S_1, S_2, \dots, S_N\}$ and a transition matrix Q. The two processes are connected via emission probabilities $b_j(k) = P[v_t = V_k | q_t = S_j]$, so $b_j(k)$ is the probability of getting the signal V_k provided the model is in state S_j.

A good overview of hidden Markov models can be found in Rabiner's tutorial on Hidden Markov models [7]. There are numerous applications of the concept in the area of pattern recognition such as speech, handwriting, gesture recognition, etc. The recent development of the topics with applications in finance is presented in [6]. There are also some applications in the area of market analysis, for instance in [5] it is suggested to describe the essential stock price movements by a hidden Markov model. In [1] portfolio optimization is considered with one bond and one stock, in [4] a utility-based portfolio selection problem is considered, where the parameters are modulated by an unobserved Markov process.

In our numerical calculations we consider the case of discrete observations, i.e., the vector of asset returns in each period should be converted into a finite discrete variable. A possible approach has been already discussed in Sect. 9.1.5, where we developed a *k-means method* for this transformation for the determination of the states of the observable Markov chain. Having a discrete variable as an observation, we need to estimate the transition matrix Q and the emission probabilities $b_j(k)$. This can be done using iterative Baum–Welch algorithm [7]. The algorithm requires some initial estimation of the transition matrix and emission probability matrix, which we obtain using the same clustering approach. To run the experiments we choose the number of possible observation values (number of the states of the observable Markov chain) as $M = 4$ and the number of hidden Markov chain states $N = 3$. In fact we apply the clustering twice—into 4 and 3 states, respectively, and then enhance the estimation of the transition matrix in the second case with the Baum–Welch algorithm (as a starting value of the transition matrix estimation).

9.2.2 Structural Analysis of Russian Stock Market

In order to understand the impact of the crisis on the structural changes of the market we fix 5 observation periods: 3 periods before crisis and 2 periods after the crisis. Then we use the Baum–Welch algorithm for each period of observation and examine the structure of the transition matrix for the hidden states for each period. As a result we observe an interesting structural change in the structure of the transition matrix and give their interpretation. The results of calculation are presented in the tables below.

Period 1 (precrisis) from 02.09.2003 to 31.01.2006:
Transition Q_1 and emission E_1 matrices for the hidden states

$$Q_1 = \begin{pmatrix} 0.63 & 0.00 & 0.37 \\ 0.00 & 0.86 & 0.14 \\ 0.34 & 0.17 & 0.49 \end{pmatrix}, \qquad E_1 = \begin{pmatrix} 0.32 & 0.68 & 0.00 & 0.00 \\ 0.00 & 0.50 & 0.50 & 0.00 \\ 0.00 & 0.00 & 0.63 & 0.37 \end{pmatrix}.$$

The limiting (stationary) probabilities for this period are given by

$$D_1 = (0.28, 0.40, 0.32).$$

Period 2 (precrisis) from 10.11.2004 to 16.04.2007:
Transition Q_2 and emission E_2 matrices for the hidden states

$$Q_2 = \begin{pmatrix} 0.61 & 0.00 & 0.39 \\ 0.04 & 0.83 & 0.13 \\ 0.21 & 0.31 & 0.48 \end{pmatrix}, \qquad E_2 = \begin{pmatrix} 0.36 & 0.64 & 0.00 & 0.00 \\ 0.00 & 0.49 & 0.51 & 0.00 \\ 0.00 & 0.00 & 0.58 & 0.42 \end{pmatrix}.$$

The limiting (stationary) probabilities for this period are given by

$$D_2 = (0.20, 0.51, 0.29).$$

Period 3 (precrisis) from 01.02.2006 to 30.06.2008:
Transition Q_3 and emission E_3 matrices for the hidden states

$$Q_3 = \begin{pmatrix} 0.54 & 0.00 & 0.46 \\ 0.16 & 0.81 & 0.03 \\ 0.16 & 0.41 & 0.43 \end{pmatrix}, \qquad E_3 = \begin{pmatrix} 0.41 & 0.59 & 0.00 & 0.00 \\ 0.00 & 0.52 & 0.48 & 0.00 \\ 0.00 & 0.00 & 0.65 & 0.35 \end{pmatrix}.$$

The limiting (stationary) probabilities for this period are given by

$$D_3 = (0.22, 0.53, 0.25).$$

Period 4 (postcrisis) from 08.12.2008 to 10.12.2010:
Transition Q_4 and emission E_4 matrices for the hidden states

$$Q_4 = \begin{pmatrix} 0.61 & 0.01 & 0.38 \\ 0.01 & 0.99 & 0.00 \\ 0.42 & 0.00 & 0.58 \end{pmatrix}, \qquad E_4 = \begin{pmatrix} 0.23 & 0.77 & 0.00 & 0.00 \\ 0.00 & 0.49 & 0.51 & 0.00 \\ 0.00 & 0.00 & 0.77 & 0.23 \end{pmatrix}.$$

The limiting (stationary) probabilities for this period are given by

$$D_4 = (0.30, 0.43, 0.27).$$

Period 5 (postcrisis) from 09.12.2009 to 14.12.2011:
Transition Q_5 and emission E_5 matrices for the hidden states

$$Q_5 = \begin{pmatrix} 0.21 & 0.55 & 0.25 \\ 0.12 & 0.46 & 0.42 \\ 0.01 & 0.37 & 0.62 \end{pmatrix}, \qquad E_5 = \begin{pmatrix} 0.99 & 0.01 & 0.00 & 0.00 \\ 0.00 & 0.77 & 0.23 & 0.00 \\ 0.00 & 0.00 & 0.67 & 0.33 \end{pmatrix}.$$

The limiting (stationary) probabilities for this period are given by

$$D_5 = (0.08, 0.42, 0.51).$$

To make an appropriate comment to the presented results we need some interpretation of the hidden states. Figure 9.7 shows the behavior of the indexes and associated hidden state. From this information we conclude that state 1 can be associated with a "jump down" of the market, state 2—with a "regular growth" of the market and state 3—with a "jump up" of the market. The same conclusion is true for the postcrisis periods. Now we can describe the peculiarity of the transition matrix for each period. For all three periods of the precrisis the limiting probabilities are nearly the same but some structural changes can be observed in the transition matrix for the hidden states. Periods 1 and 2 are marked by the very low probability of transition from the state 1 (jump down) to state 2 (regular growth) and vise versa. This structure is perturbed near the crisis period. Period 3 is marked by the very low probability of transition from state 1 (jump down) to state 2 (regular growth) and from state 2 (regular growth) to state 3 (jump up). It means that the market became less regular looking on the hidden states. The first postcrisis period, period 4, is marked by an interesting phenomena: state 2 (regular growth) became the most attractive and the probabilities of transition from this state to states 1 (jump down) and 3 (jump up) are very low. However this phenomenon is not translated into the limiting distribution which is closed to be uniform. Finally the period 5 is marked by

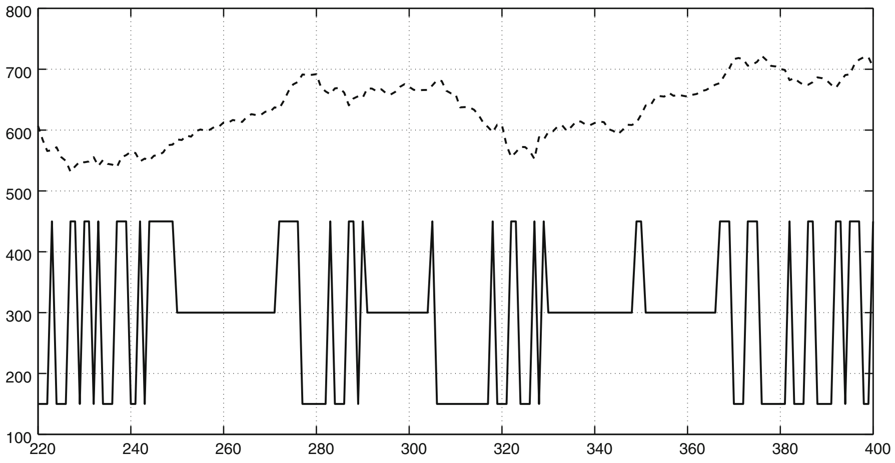


Fig. 9.7 Market behavior and hidden states before crisis

the very low probability of transition from state 3 (jump up) to state 1 (jump down). It means that the market is good for the investment. This conclusion is confirmed by the limiting distribution of the state probabilities. This distribution is shifted to the favorable states 2 and 3.

9.3 Conclusion

In the present chapter we used Markov chain models for the analysis of some problems for the Russian stock market. Two problems were addressed. One is concerned with the multiperiod portfolio optimization. Our finding is a nonstability phenomena for the optimal portfolio related with the choice of the states of the Markov model. Second problem deals with the structural changes on the market due to the financial crisis of 2008. Using the hidden Markov chain model we detected some interesting dynamics of the structure of the hidden states transition matrix over the crisis dynamics.

Acknowledgements The authors acknowledge the RF Grant No. 11.G34.31.0057 for the support.

References

1. N. Bäuerle, U. Rieder, Portfolio optimization with jumps and unobservable intensity process. Math. Finance **17**(2), 205–224 (2007)
2. N. Bäuerle, U. Rieder, *Markov Decision Processes with Applications to Finance* (Springer, Berlin, 2011)

3. U. Çakmak, S. Özekici, Portfolio optimization in stochastic markets. Math. Meth. Oper. Res. **63**(1), 151–168 (2006)
4. E. Çanakoğlu, S. Özekici, Portfolio selection in stochastic markets with exponential utility functions. Ann. Oper. Res. **166**(1), 281–297 (2009)
5. R. Elliott, Y. Hinz, Portfolio optimization, hidden Markov models, and technical analysis of P&F-Charts. Int. J. Theor. Appl. Finance **5**(4), 385–399 (2002)
6. R.S. Mammon, R.J. Elliott, *Hidden Markov Models in Finance* (Springer, New York, 2007)
7. L.R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE **77**(2), 257–286 (1989)
8. L. Stettner, Risk-sensitive portfolio optimization with completely and partially observed factors. IEEE Trans. Automat. Contr. **49**, 457–464 (2004)

Chapter 10
Fuzzy Portfolio Selection Models: A Numerical Study

Enriqueta Vercher and José D. Bermúdez

Abstract In this chapter we analyze the numerical performance of some possibilistic models for selecting portfolios in the framework of risk-return trade-off. Portfolio optimization deals with the problem of how to allocate wealth among several assets, taking into account the uncertainty involved in the behavior of the financial markets. Different approaches for quantifying the uncertainty of the future return on the investment are considered: either assuming that the return on every individual asset is modeled as a fuzzy number or directly measuring the uncertainty associated with the return on a given portfolio. Conflicting goals representing the uncertain return on and risk of a fuzzy portfolio are analyzed by means of possibilistic moments: interval-valued mean, downside-risk, and coefficient of skewness. Thus, several nonlinear multi-objective optimization problems for determining the efficient frontier could appear. In order to incorporate possible trading requirements and investor's wishes, some constraints are added to the optimization problems, and the effects of their fulfillment on the corresponding efficient frontiers are analyzed using a data set from the Spanish stock market.

10.1 Introduction

The portfolio selection problem deals with finding an optimal strategy for building satisfactory portfolios. From Markowitz's seminal work many different modeling approaches have been developed in order to propose suitable investment strategies [44]. Concerning uncertainty quantification, it is assumed that these decision problems can be modeled either by using the stochastic tools provided by probability theory or by using soft computing approaches based on fuzzy set theory [65].

E. Vercher (✉) • J.D. Bermúdez
Department of Statistics and Operational Research, University of Valencia,
C/ Dr. Moliner 50, 46100-Burjassot, Spain
e-mail: vercher@uv.es; bermudez@uv.es

The decision problem under uncertainty must be then approached using different mathematical tools, and from an optimization point of view both linear and nonlinear programming and different meta-heuristic techniques can be used for solving the portfolio selection problem.

Modern portfolio selection theory usually deals with two opposite concepts: risk aversion and maximizing return. The main point of this modeling approach is how risk and asset profitability are defined and measured. Following Markowitz's proposal classic models consider the return on an asset as a random variable and its profitability is defined as the mathematical expectation of that random variable, while risk is measured by means of the variance. Since the formulation of the mean–variance model, a variety of enlarged and improved models have been developed in several directions. One dealt with alternative portfolio selection models either by modifying the risk measure (mean–semivariance model [45], mean-absolute deviation model [31], mean–downside risk models [54, 55]) or by adding higher probability moments to best represent the uncertainty on the returns [30,34]. Others dealt with the introduction of factors influencing stock prices based on the Capital Asset Pricing Model [42,50] and derivative methodologies.

In order to identify the best portfolio for a given level of desired return, Levy and Markowitz [39] propose extending the classic portfolio selection models into multi-criteria decision-making models. A straightforward approach to selecting an optimal portfolio is minimizing the risk and maximizing the expected return simultaneously, that is considering a bi-objective optimization problem. From then on, multi-objective programming techniques have been applied for portfolio selection, and the solutions are usually obtained using scalar optimization by aggregation of the objectives into a single one [3,57]. Some multi-objective approaches also allow the incorporation of higher-order moments as several alternative criteria for portfolio selection [10,34,56,62].

Another approach to uncertainty quantification is based on fuzzy set theory, which also provides a framework for the analysis of decisions about investment under imperfect knowledge of future market behavior. Different elements and characteristics of the portfolio selection problem can be fuzzified such as the following papers reflect. Watada [63] uses fuzzy sets to introduce the vague goals of the decision makers for the expected rate of return and risk in the mean–variance model. Ramaswamy [48] applies fuzzy decision theory to selecting optimal portfolios with targets above the risk-free rate, taking into account only market risk under different scenarios of market behavior. Tanaka and Guo [58] use possibility distributions to model uncertainty in the returns, allowing the incorporation of expert knowledge by means of a possibility grade, which reflects the degree of similarity between the future state of stock markets and the state of previous periods. Arenas et al. [2] propose a fuzzy goal programming approach for portfolio selection based on a factor model, also taking into account the liquidity of the investment. León et al. [37] present a fuzzy interactive algorithm for selecting optimal portfolios that uses a modification of Zimmermann's method [67] for solving multi-objective decision problems; the authors also describe a fuzzy optimization scheme to manage unfeasible instances of the portfolio selection problem.

In this chapter we analyze the performance of certain possibilistic multi-objective portfolio selection models with the main objective of providing the investor with sufficient tools to address the portfolio selection problem. Discrete constraints representing trading requirements and investors' preferences are introduced by means of diversification and restricted cardinality conditions. The optimization scheme goals are to minimize the risk of the investment while maximizing the expected return and skewness. Then, a multi-objective evolutionary algorithm finds efficient portfolios that also meet the investors' wishes. The uncertainty in the expected income is represented by modeling the returns on the individual assets as trapezoidal fuzzy numbers, or alternatively, directly computing the trapezoidal fuzzy number that approximates to the return on a given portfolio, without requiring the estimation of the joint distribution of the returns on the assets. The expected return on the investment are approximated by using interval-valued possibilistic means, the risk being measured as a fuzzy downside risk. In order to analyze whether the introduction of skewness would significantly improve the quality of chosen portfolios, the relationship between the fuzzy downside risk and the possibilistic coefficient of skewness is considered, also with respect to the observation that increasing diversification could lead to a loss of skewness. Our approaches for selecting efficient portfolios are applied to a selection of assets from the Spanish Stock Market.

The rest of the chapter is organized in the following manner. The next section presents a brief summary of portfolio optimization models. In Sect. 10.3 we include some definitions and basic results of fuzzy sets and possibilistic moments, which are used to represent the uncertainty of returns. Section 10.4 describes the formulation of the portfolio selection problem as a possibilistic multi-objective programming problem. Section 10.5 shows that a satisfactory solution can be easily obtained using a meta-heuristic procedure. Suitable comparisons are reported, also showing the influence of the presence of discrete constraints and the effect of considering skewness as an effective goal of the optimization scheme. Section 10.6 gives the conclusions.

10.2 Portfolio Optimization Models

Let us present some portfolio modeling approaches, paying special attention to the optimization techniques that have been used for selecting optimal portfolios.

10.2.1 Portfolio Selection Models Based on Probability Theory

Markowitz shows how rational investors can construct optimal portfolios under conditions of uncertainty using both probability theory and optimization techniques. In the mean–variance (MV) portfolio approach the return on any portfolio is quantified as its expected value and its risk is quantified as its variance. In a standard formulation we have the following quadratic programming problem, for a given expected return ρ:

$$\begin{array}{ll} \text{(MV) Min} & \sum_{j=1}^{n}\sum_{i=1}^{n}\sigma_{ij}x_ix_j \\ \text{s.t.} & \sum_{j=1}^{n}x_jE(R_j)=\rho \\ & \sum_{j=1}^{n}x_j=1 \\ & x_j\geq 0, \end{array}$$

where x_j is the proportion of the portfolio held in the asset jth, R_j is the random variable representing the return on asset jth, and σ_{ij} is the covariance between R_i and R_j, for $i,j=1,\dots,n$. The solution for this quadratic program for different values of ρ allows identifying the set of efficient portfolios (see [4], for a complete report on nonlinear optimization). It is usual to plot the pairs (σ,ρ) to represent the portfolio efficient frontier where σ is the standard deviation.

A number of researchers have introduced alternative measures of risk for portfolio planning, and in many cases these measures are linear, leading to a corresponding simplification in the computational model. According to Konno and Yamazaki [31], when the returns on the assets are multivariate normally distributed, the above portfolio selection problem is equivalent to the mean-absolute deviation model (MAD), which minimizes the sum of absolute deviations from the averages associated with the x_j choices:

$$\begin{array}{ll} \text{(MAD) Min} & E(|\sum_{j=1}^{n}R_jx_j-E(\sum_{j=1}^{n}R_jx_j)|) \\ \text{s.t.} & \sum_{j=1}^{n}x_jE(R_j)\geq M\rho \\ & \sum_{j=1}^{n}x_j=M \\ & 0\leq x_j\leq u_j, \end{array}$$

where M is the total fund and u_j represents the maximum amount of the total fund which can be invested in the asset jth. This modeling approach permits avoidance of one of the main drawbacks associated with the solution of the MV model: the input problem of estimating $2n+n(n-1)/2$ parameters. Moreover, it can be easily converted into a finite linear optimization problem by replacing its objective function with:

$$\begin{array}{lll} \text{Min} & (1/T)\sum_{k=1}^{T}y_k & \\ \text{s.t.} & y_k+\sum_{j=1}^{n}(r_{kj}-E(R_j))x_j\geq 0 & k=1,\dots,T \\ & y_k-\sum_{j=1}^{n}(r_{kj}-E(R_j))x_j\geq 0 & k=1,\dots,T, \end{array}$$

where the returns on the assets over T periods are given and r_{kj} denotes the return on the jth asset at the time k, for $k=1,\dots,T$. The corresponding linear programming problem (LMAD) gives portfolios which involve fewer nonzero components and hence reduces the numerous small transactions that are likely to appear in the MV model.

Concerning the performance of the above modeling approaches, Simaan [51] stated that although the minimization of mean-absolute deviation is close to the MV formulation, they lead to two different efficient sets. This divergence is probably due to the fact that each model utilizes different sample statistics and consequently relies on a different set drawn from the sample. On the other hand, since the normality assumption is rarely verified in practice, different portfolios can be obtained when the above models are applied. Júdice et al. [29] use MV and LMAD models in real-life capital markets for analyzing the stability of the selected portfolios and their expected return, and conclude that no one model is superior to the other. Papahristodoulou and Dotzauer [47] also compare them with out-of-sample data from shares traded in the Stockholm Stock Exchange and the results show that the MV model yields higher utility levels and higher degrees of risk aversion in very similar computing times.

Dissatisfaction with the traditional notion of variance as a measure of investment risk may be due to the fact that it makes no distinction between gains and losses. Thus, Markowitz [45] proposed the use of the semi-variance instead of the variance:

$$\mathrm{SV}(x) = \mathrm{E}\left(\left(\max\left\{0, \mathrm{E}(\sum_{j=1}^{n} R_j x_j) - \sum_{j=1}^{n} R_j x_j\right\}\right)^2\right) \tag{10.1}$$

From then on, several optimization models which consider only the downside risk of a portfolio have been introduced. In particular, if risk is measured by means of the mean-absolute semi-deviation, as proposed in Speranza [55], the following risk function appears:

$$\mathrm{SD}(x) = \mathrm{E}\left(\left|\min\left\{0, \sum_{j=1}^{n} R_j x_j - \mathrm{E}(\sum_{j=1}^{n} R_j x_j)\right\}\right|\right) \tag{10.2}$$

which can be easily evaluated in contrast with the complexity of computing the semi-variance of a given portfolio. Konno et al. [32] review the performance of different lower partial measures of risk by applying linear programming techniques, which also allows the efficient resolution of large-scale instances.

Recently, a lot of research has been undertaken with the purpose of constructing efficient portfolios using meta-heuristic techniques [1, 14, 43] and for developing decision support system (DSS) strategies for assessing the investors during the process of decision making [64, 69].

10.2.2 Fuzzy Approaches for Portfolio Selection

Portfolio selection models use parameters, goals, and constraints whose characteristics and values are imprecise in a certain sense. This imperfect knowledge can be introduced by means of fuzzy quantities in very different ways. Let us mention some of them.

The portfolio selection problem under uncertainty can be transformed into a problem of decision making in a fuzzy environment by modeling the investors' aspiration levels for the expected return on and risk of through suitable membership functions of a fuzzy set. For instance, the fuzzy portfolio selection model in Watada [63], based on the mean–variance model, assigns a logistic membership function to the goal of expected return, as follows:

$$\mu_E\left(\sum_{j=1}^{n} x_j E(R_j)\right) = \frac{1}{1+\exp[-\beta_1(\sum_{j=1}^{n} x_j E(R_j) - E_M)]}, \tag{10.3}$$

where β_1 is the positive shape parameter of the logistic function and E_M is the point in the support of the membership function with a value of 0.5 (analogously for the membership function of the goal associated with risk: β_2 and V_M). Since goals and constraints can be represented as a fuzzy set, the corresponding decision problem can be defined as the intersection of goals and constraints. Therefore, if we consider a full compensation approach, it would be more appropriate to define the membership function of the fuzzy decision by taking the maximum degree of membership achieved by any of the fuzzy sets representing objectives or constraints. The portfolio selection problem can then be stated as

$$\begin{aligned}
\text{(FMV) Max } & \lambda \\
\text{s.t.} \quad & \lambda + (\exp[-\beta_1(\textstyle\sum_{j=1}^{n} x_j E(R_j) - E_M)])\lambda \le 1 \\
& \lambda + (\exp[\beta_2(\textstyle\sum_{j=1}^{n} \sigma_{ij} x_i x_j - V_M)])\lambda \le 1 \\
& \textstyle\sum_{j=1}^{n} x_j = 1 \\
& x_j \ge 0,
\end{aligned}$$

where λ is the degree of satisfaction of the solution of the above nonlinear programming problem.

Another approach to managing uncertainty is based on possibility distributions which are associated with fuzzy variables [20, 66]. Tanaka and Guo [58], for instance, use exponential possibility distribution to build a possibility portfolio model which integrates the historical data set of returns on individual assets and experts' experience and judgment. Two possibility distributions (upper and lower) are identified from the given possibility degrees for data that reflect two opposing expert opinions. Portfolio selection problems based on upper and lower possibility distributions are formalized as quadratic programming problems minimizing the spreads of possibility portfolios subject to the given center returns (r_c), as follows:

$$\begin{aligned}
\text{(TG) Min } & x^t D_A x \\
\text{s.t.} \quad & c^t x = r_c \\
& \textstyle\sum_{j=1}^{n} x_j = 1 \\
& x_j \ge 0.
\end{aligned}$$

The estimation of the matrix D_A is made using a linear programming problem, where the data are (r_k, h_k), $k = 1, \ldots, T$ and $r_k = (r_{1k}, \ldots, r_{nk})^t$ is the vector of the returns on n assets during the kth period, h_k being an associated possibility grade given by expert knowledge to reflect the degree of similarity between the future state of stock markets and the state of the *k*th sample. The authors show that the spread of the portfolio return based on a lower possibility distribution is smaller than the spread of the portfolio return based on an upper possibility distribution for the same center value. More information about this possibilistic model and other approaches to portfolio selection based on fuzzy set theory can be found in [22].

In contrast with the above approaches other authors propose the incorporation of fuzzy numbers to directly represent the uncertainty of the future returns on the assets [12,38,46]. In Carlsson et al. [12], the rates of return on securities are modeled by possibility distributions, and the return on, $E(r_p)$, and risk of the portfolio $\sigma^2(r_p)$ are, respectively, quantified using the possibilistic mean and variance previously defined in [11]. The authors find an exact optimal solution to the following portfolio selection problem under trapezoidal possibility distributions:

$$\begin{aligned}\text{(CFM) Max } & E(r_p) - 0.005A\sigma^2(r_p)\\ \text{s.t. } & \textstyle\sum_{j=1}^{n} x_j = 1\\ & x_j \geq 0,\end{aligned}$$

where *A* is an index of the investors' aversion to risk. Alternatively, assuming that the downside risk is a more realistic description of investor preferences, because this risk function only penalizes the non-desired deviations, some fuzzy models for portfolio selection have been proposed [61]. The fuzzy downside risk function evaluates the mean-absolute semi-deviation with respect to the total return $\tilde{R}_P(x)$ as follows [38]:

$$\mathrm{w}_P(x) = \mathrm{E}(\max\{0, \mathrm{E}(\tilde{R}_P(x)) - \tilde{R}_P(x)\}) \tag{10.4}$$

where the total return on the fuzzy portfolio is a convex linear combination of the individual asset returns, that is: $\tilde{R}_P(x) = \sum_{j=1}^{n} x_j \tilde{R}_j$ and $\tilde{R}_j$ are LR-fuzzy numbers, for $j = 1, 2, \ldots, n$. Then, the fuzzy mean–downside risk portfolio selection problem can be stated in the following way:

$$\begin{aligned}\text{(MDR) Min } & \mathrm{E}(\max\{0, \mathrm{E}(\tilde{R}_P(x)) - \tilde{R}_P(x)\})\\ \text{s.t. } & \textstyle\sum_{j=1}^{n} x_j \tilde{R}_j \succeq \tilde{r}\\ & \textstyle\sum_{j=1}^{n} x_j = 1\\ & l_j \leq x_j \leq u_j,\end{aligned}$$

where $l_j \geq 0$ represents the minimum amount of the total fund which can be invested in asset *j*th. The calculation of the fuzzy expected return and downside risk depends on both the characteristics of the LR-fuzzy numbers which represent the individual returns and the definition of the average of a fuzzy number. Also, for satisfying the fuzzy constraint, different approaches can be used (see, for instance, [59]). Using interval-valued expectations the above MDR model provides optimal portfolios by applying linear optimization, linear semi-infinite optimization, or linear interval programming, for different modeling approaches.

Recently, Huang [27] has introduced fuzzy models for portfolio selection under the assumption that the returns on the assets are random fuzzy variables; other approaches which deal with randomness and fuzziness simultaneously have also been proposed (see, for instance, [25, 33]).

The next section introduces definitions of and criteria for modeling return and risk using possibilistic moments and then looks into other related criteria which are based on fuzzy logic to represent the uncertainty of future returns on assets and portfolios.

10.3 Modeling Uncertainty

The concept of fuzzy sets was introduced by Zadeh in 1965, and since then it has been used for modeling uncertainty or impreciseness in data. Fuzzy set theory allows a more precise mathematical description to be given of what are normally vague statements, and has become an interesting tool when applied to decision problems (see, for instance, [5, 52, 68]). A deep and comprehensive treatment of fuzzy sets and their properties is provided, for instance, in [21].

10.3.1 Fuzzy Numbers and Fuzzy Arithmetic

Let us briefly recall some definitions and results which will be used in what follows.

Definition 10.1. Let X denote the universal set. A fuzzy set $\tilde{A}$ in X is characterized by a membership function $\mu_{\tilde{A}}(x)$ which associates a real number in the interval [0,1] with each point in X, where the value of $\mu_{\tilde{A}}(x)$ at x represents the *grade of membership* of x in $\tilde{A}$.

Definition 10.2. A fuzzy number $\tilde{A}$ is a fuzzy set defined on the set of real numbers $\Re$, characterized by means of a membership function $\mu_{\tilde{A}}(x)$ which is upper semi-continuous and satisfies the condition $\sup_{x\in\Re}\mu_{\tilde{A}}(x) = 1$, and whose α-cuts, for $0 \leq \alpha \leq 1$: $[\tilde{A}]^{\alpha} = \{x \in \Re : \mu_{\tilde{A}}(x) \geq \alpha\}$, are convex sets.

Definition 10.3. A fuzzy number is said to be a trapezoidal LR-fuzzy number, $\tilde{A} = (a_l, a_u, c, d)_{LR}$, if its membership function has the following form:

$$\mu_{\tilde{A}}(y) = \begin{cases} L(\frac{a_l-y}{c}) = \frac{y-(a_l-c)}{c} & \text{if } y \in [a_l - c, a_l] \\ 1 & \text{if } y \in [a_l, a_u] \\ R(\frac{y-a_u}{d}) = \frac{a_u+d-y}{d} & \text{if } y \in [a_u, a_u + d] \\ 0 & \text{otherwise} \end{cases},$$

where the reference functions $L,R : [0,1] \to [0,1]$ are linear, $[a_l,a_u]$ is the core of $\tilde{A}$, and the closure of the support of $\tilde{A}$, $\mathrm{supp}(\tilde{A}) = \{y : \mu_{\tilde{A}}(y) > 0\}$, is exactly $[a_l - c, a_u + d]$.

The aggregation and ranking of positive linear combinations of LR-fuzzy numbers have been extensively dealt with when their reference functions are linear or all of them have the same shape, because it provides LR-fuzzy numbers of the same shape. Thus, using Zadeh's extension principle, the following arithmetical rules hold.

Theorem 10.1. *Let $\tilde{A} = (a_l,a_u,c_1,d_1)_{LR}$ and $\tilde{B} = (b_l,b_u,c_2,d_2)_{LR}$ be two LR-fuzzy numbers and let $\lambda \in \Re$ be a real number. Then,*

1. $\tilde{A} + \tilde{B} = (a_l + b_l, a_u + b_u, c_1 + c_2, d_1 + d_2)_{LR}$

2. $\lambda\tilde{A} = \begin{cases} (\lambda a_l, \lambda a_u, \lambda c_1, \lambda d_1)_{LR} & if\ \lambda \geq 0 \\ (\lambda a_u, \lambda a_l, |\lambda| d_1, |\lambda| c_1)_{LR} & if\ \lambda < 0 \end{cases}$,

where the addition and multiplication by a scalar is defined by the sup-min extension principle.

The above result cannot be applied to differently shaped LR-fuzzy numbers, where this aggregation is defined with respect to the α-level sets of the fuzzy number $\tilde{A}$ [20, 36].

10.3.2 Possibilistic Moments

The possibility measure of an event might be interpreted as the possibility degree of its occurrence under a possibility distribution. Let $\tilde{A}$ be a fuzzy number, the membership function values for every $x \in \Re$ can be interpreted as the degree of possibility of the statement "x is the value of $\tilde{A}$." Since the fuzzy numbers can be considered a special class of possibility distributions, the imprecise coefficients and vagueness and imprecision of data may be modeled by means of possibility distributions [66].

In this chapter the uncertainty regarding the returns of a given investment is modeled by means of fuzzy quantities for which different definitions of the average can be used. Let us remember some of them. In 1987 Dubois and Prade defined the mean value of a fuzzy number as a closed interval bounded by the expectations calculated from its lower and upper probability mean values [19]:

Definition 10.4. The interval-valued expectation of a fuzzy number $\tilde{A}$ is the following interval: $E(\tilde{A}) = [E_*(\tilde{A}), E^*(\tilde{A})]$, whose endpoints are $E_*(\tilde{A}) = \int_0^1 \mathrm{inf}\tilde{A}_\alpha \mathrm{d}\alpha$ and $E^*(\tilde{A}) = \int_0^1 \mathrm{sup}\tilde{A}_\alpha \mathrm{d}\alpha$, where $\mathrm{inf}\tilde{A}_\alpha$ and $\mathrm{sup}\tilde{A}_\alpha$ denote the left and right extreme points of the α-cut of $\tilde{A}$ for $0 \leq \alpha \leq 1$.

This mean interval definition provides the nearest interval approximation to the fuzzy number with respect to the metric introduced in [24], its width being equal to the width of $\tilde{A}$ in the Chanas sense [13]. We will use the midpoint of this mean interval as the crisp representation of the fuzzy expected return, which is denoted as $\bar{E}(\tilde{A})$. In order to incorporate the importance of α-level sets into the definition of mean value of a fuzzy quantity, Fullér and Majlender [23] introduce the concept of weighted possibilistic expectation of fuzzy numbers, extending the definition of interval-valued possibilistic mean and variance given by Carlsson and Fullér [11].

Definition 10.5. Let $\tilde{A}$ be a fuzzy number and $f(\alpha)$ be a weighted function. The f-weighted possibilistic mean of $\tilde{A}$ is the following interval: $M(\tilde{A}) = [M_*(\tilde{A}), M^*(\tilde{A})]$, where the endpoints are calculated as $M_*(\tilde{A}) = \int_0^1 f(\alpha)\inf\tilde{A}_\alpha \mathrm{d}\alpha$ and $M^*(\tilde{A}) = \int_0^1 f(\alpha)\sup\tilde{A}_\alpha \mathrm{d}\alpha$.

A variety of alternative criteria for the definition of higher order moments for the portfolio selection can be found in the literature. Recently, Saedifar and Pasha [49] have introduced new weighted possibilistic moments of fuzzy numbers and analyzed the properties of the nearest weighted possibilistic points which are usually used in the stage of defuzzification processes. Let us recall their definition of a 3rd weighted possibilistic moment, which will be useful in the context of measuring the asymmetry of LR-fuzzy numbers.

Definition 10.6. Let $\tilde{A}$ be a fuzzy number and $f(\alpha)$ be a weighted function. The 3rd weighted possibilistic moment about the weighted possibilistic mean value of $\tilde{A}$, $\bar{M}(\tilde{A})$, is

$$\mu_3(\tilde{A}) = \frac{1}{2}\int_0^1 f(\alpha)[\inf\tilde{A}_\alpha - \bar{M}(\tilde{A})]^3 \mathrm{d}\alpha + \frac{1}{2}\int_0^1 f(\alpha)[\sup\tilde{A}_\alpha - \bar{M}(\tilde{A})]^3 \mathrm{d}\alpha \quad (10.5)$$

where $\bar{M}(\tilde{A})$ is the midpoint of the f-weighted possibilistic mean interval.

Throughout this chapter we work with the interval-valued expectation given in Definition 10.4, which means that we assume that all the α-cuts have the same weight: $f(\alpha) = 1$ for every $0 \leq \alpha \leq 1$, although the introduction of different weights for every α should be straightforward.

10.3.3 Fuzzy Risk and Expected Return on Portfolios

Let us consider a portfolio in which the total wealth has to be allocated to n risky assets. The vector $X = (x_1, \ldots, x_n)$ represents the proportions of the total investment devoted to each asset j^{th}, for $j = 1, \ldots, n$, and the components of vector X are restricted to the basic constraints of $\sum_{j=1}^n x_j = 1$ and $x_j \geq 0$, if the short-selling of the assets is not allowed. The uncertainty regarding its future return is modeled by means of fuzzy quantities based on the historical returns over T periods: $\{r_{tj}\}$, for $t = 1, \ldots, T$ and $j = 1, \ldots, n$. Therefore, associated with each rate of return there is a possibility distribution defined by the membership function of the corresponding fuzzy set. This approach allows the application of the above definitions of possibilistic moments in order to quantify the uncertainty associated with the future return on a portfolio, X.

Here we will analyze two different approaches to the measurement of the uncertainty on the return of a risky investment. Firstly, modeling the return on the individual assets using trapezoidal LR-fuzzy numbers $\tilde{R}_j$ and alternatively considering the returns on a given portfolio as the historical data set.

10.3.3.1 Fuzzy Returns on the Individual Assets

Let us denote the return on the jth asset by $\tilde{R}_j = (a_{lj}, a_{uj}, c_j, d_j)_{LR}$, a trapezoidal fuzzy number whose α-level cuts are $[\tilde{R}_j]^\alpha = [a_{lj} - c_j(1-\alpha), a_{uj} - dj(1-\alpha)]$, for $\alpha \in [0,1]$. The core and spreads of the fuzzy return on every asset j_0 are computed as functions of the sample percentiles of the data in the corresponding column: $\{r_{tj_0}\}_{t=1}^{T}$. Then, the total fuzzy return on the portfolio X is the following trapezoidal fuzzy number:

$$\begin{aligned}\tilde{R}_P(x) &= \textstyle\sum_{j=1}^n x_j \tilde{R}_j = \left(\sum_{j=1}^n a_{lj}x_j, \sum_{j=1}^n a_{uj}x_j, \sum_{j=1}^n c_j x_j, \sum_{j=1}^n d_j x_j\right)_{LR} \\ &= (P_l(x), P_u(x), C(x), D(x))_{LR}.\end{aligned}$$

This approach does not require the estimation of the joint possibility distribution of the return on the assets, which is not usually computable with any degree of confidence.

Since the interval-valued expectation remains additive in the sense of the addition of fuzzy numbers (Definition 10.4), we can easily compute the possibilistic moments of the total fuzzy return $\tilde{R}_P(x)$. It is easy to see that its interval-valued mean is

$$E(\tilde{R}_P(x)) = \left[P_l(x) - \tfrac{C(x)}{2}, P_u(x) + \tfrac{D(x)}{2}\right] \tag{10.6}$$

then as a scalar representative of this mean-interval we use its middle point:

$$\bar{E}(\tilde{R}_P(x)) = \frac{1}{2}\sum_{j=1}^n (a_{lj} + a_{uj})x_j + \frac{1}{4}\sum_{j=1}^n (d_j - c_j)x_j. \tag{10.7}$$

In terms of measuring the risk of the investment, we will use the fuzzy downside risk defined in (10.4). Then, applying the interval-valued expectation given in Definition 10.4, it can be found that this fuzzy interval risk is (see [61] for details):

$$\mathrm{w}_P(x) = \left[0, P_u(x) - P_l(x) + \frac{1}{3}(C(x) + D(x))\right] \tag{10.8}$$

and we use the length of this interval mean as a crisp representation of the investment risk:

$$\bar{\mathrm{w}}_P(x) = \sum_{j=1}^n \left(a_{uj} - a_{lj} + \frac{1}{2}(c_j + d_j)\right) x_j. \tag{10.9}$$

Delgado et al. [18], in the context of ranking fuzzy numbers, define the value $V(\tilde{B})$ and ambiguity $A(\tilde{B})$ of a fuzzy number $\tilde{B}$. It is easy to see that the value of $\tilde{R}_P(x)$, with respect to the reducing function $s(r) = r$, coincides with the expected return given by (10.7). Moreover, (10.9) is twice the ambiguity of $\tilde{R}_P(x)$. This fact reinforces the idea that these values are picking up the inexactness of the future return and they could be useful for selecting suitable sharing portfolios.

When the return on the portfolio is not symmetrically distributed around the mean, it is usually recommended to measure this asymmetry by using information about the 3rd moment of the possibility distribution. On the other hand, to obtain a relative measure of the asymmetry of the returns on fuzzy portfolios, we have recently introduced the following definition [60]:

Definition 10.7. Let $\tilde{R}_P(x)$ be the total return on a portfolio and $\mu_3(\tilde{R}_P(x))$ its 3rd possibilistic moment about the mean value $\bar{E}(\tilde{R}_P(x))$, then the coefficient of possibilistic skewness of $\tilde{R}_P(x)$ is defined as

$$S(\tilde{R}_P(x)) = \frac{\mu_3(\tilde{R}_P(x))}{(\bar{\mathrm{w}}_P(x))^3} \tag{10.10}$$

For trapezoidal fuzzy numbers we have also proved that the above coefficient of skewness can be calculated by means of the following ratio:

$$S(\tilde{R}_P(x)) = \frac{1}{16}\frac{D(x)^2 - C(x)^2}{(\bar{\mathrm{w}}_P(x))^2} \tag{10.11}$$

The values in (10.7) and (10.9), which are the crisp representation of the possibilistic expected return and downside risk of the total fuzzy return on a portfolio, have previously been used to define the goals and constraints of portfolio selection problems with fuzzy returns [38, 61].

10.3.3.2 Fuzzy Return on a Given Portfolio

The above approach does not incorporate the contemporary relationship of the returns on the individual assets into the portfolio composition, because their historical information has been independently analyzed. Since our main interest is to suitably model the returns on a given portfolio, we can directly consider its returns as the historical data set, instead of considering the individual returns on the assets as the data set [6, 7], therefore the main difference between these approaches appears in the modeling of uncertainty.

Now we propose to model the uncertainty about the future returns on a given portfolio $X = (x_1, \dots, x_n)$ by using the information provided by the data set: $\{r_t(X)\}_{t=1}^T$, in such a way that the contemporary relationship among the individual returns is considered for each period $t = 1, \dots, T$. We define this contemporary return on X as follows:

$$r_t(X) = \sum_{j=1}^{n} r_{tj} x_j. \tag{10.12}$$

Table 10.1 Yearly returns on five securities (1937–1954) from Markowitz's historical data

Year	Am. T.	A.T.T.	U.S.S.	C.C.	Frstn.
1937	−0.305	−0.173	−0.318	−0.065	−0.400
1938	0.513	0.098	0.285	0.238	0.336
1939	0.055	0.200	−0.047	−0.078	−0.093
1940	−0.126	0.030	0.104	−0.077	−0.090
1941	−0.280	−0.183	−0.171	−0.187	−0.194
1942	−0.003	0.067	−0.039	0.156	0.113
1943	0.428	0.300	0.149	0.351	0.580
1944	0.192	0.103	0.260	0.233	0.473
1945	0.446	0.216	0.419	0.349	0.229
1946	−0.088	−0.046	−0.078	−0.209	−0.126
1947	−0.127	−0.071	0.169	0.355	0.009
1948	−0.015	0.056	−0.035	−0.231	0.000
1949	0.305	0.038	0.133	0.246	0.223
1950	−0.096	0.089	0.732	−0.248	0.650
1951	0.016	0.090	0.021	−0.064	−0.131
1952	0.128	0.083	0.131	0.079	0.175
1953	−0.010	0.035	0.006	0.067	−0.084
1954	0.154	0.176	0.908	0.077	0.756

The sample percentiles of this data set define the core and spreads of the trapezoidal fuzzy number $\tilde{X} = (p_l, p_u, c, d)_{LR}$, which represents the uncertainty about the future returns on X. Then, we have the same possibilistic model and its measures of risk and return are obtained by using the corresponding definitions. Then, we have:

$$\bar{E}(\tilde{X}) = \frac{p_l + p_u}{2} + \frac{d-c}{4} \tag{10.13}$$

$$\bar{w}(\tilde{X}) = p_u - p_l + \frac{d+c}{2} \tag{10.14}$$

$$S(\tilde{X}) = \frac{1}{16}\frac{d^2 - c^2}{\bar{w}(\tilde{X})^2}. \tag{10.15}$$

Let us show the performance of these two fuzzy approaches for modeling uncertainty by using the set of historical data introduced by Markowitz [45].

10.3.3.3 Numerical Example

Let us assume that an investor wants to distribute one unit of wealth among five securities (for instance: American Tobacco, A.T.T., United States Steel, Coca-Cola, and Firestone) from the Markowitz data set. Table 10.1 shows their yearly returns $\{r_{tj}\}$ from 1937 to 1954, for $t = 1,\ldots,18$ and $j = 1,\ldots,5$.

Table 10.2 Possibilistic moments of six portfolios built from Markowitz's historical data

Portfolio	$\bar{E}(\tilde{R}_P(x))$	$\bar{w}_P(x)$	$S(\tilde{R}_P(x))$	$\bar{E}(\tilde{X})$	$\bar{w}(\tilde{X})$	$S(\tilde{X})$
$X_1 = (0.1, 0.1, 0.4, 0.2, 0.2)$	0.121	0.552	0.041	0.108	0.483	0.030
$X_2 = (0.15, 0.15, 0.35, 0.2, 0.15)$	0.111	0.520	0.040	0.094	0.440	0.021
$X_3 = (0.1, 0.2, 0.3, 0.3, 0.1)$	0.099	0.481	0.036	0.089	0.383	0.010
$X_4 = (0.15, 0.25, 0.25, 0.25, 0.1)$	0.094	0.463	0.034	0.076	0.359	0.006
$X_5 = (0.1, 0.4, 0.2, 0.2, 0.1)$	0.090	0.427	0.028	0.075	0.322	−0.005
$X_6 = (0.35, 0.2, 0.15, 0.15, 0.15)$	0.089	0.469	0.031	0.064	0.370	−0.007

Table 10.2 shows the performance of certain portfolios for the above approaches to modeling uncertainty on future return. Firstly, for each asset j the core of the trapezoidal fuzzy returns is approximated using the interval $[q_{40}, q_{60}]$, q_k being the kth percentile of the sample $\{r_{tj}\}_{t=1}^{18}$, where the support is the interval between the minimum and maximum observed return. Once each trapezoidal fuzzy number $\tilde{R}_j$ has been built, we can evaluate the total fuzzy return $\tilde{R}_{P_i}(x)$ for each portfolio $\{X_i\}$ and their possibilistic moment values. On the other hand, we can also approximate the fuzzy returns on $\tilde{X}_i$ by using the same percentiles for the core and spreads from the sample $\{r_t(X)\}$ and explicitly evaluate their possibilistic moment values using (10.13)–(10.15).

The first columns in Table 10.2 show the possibilistic moments for the total return on the portfolios, by assuming fuzzy returns on individual assets, and the last three ones for the direct evaluation of the returns on given portfolios. Note that the value of possibilistic moments is usually greater when the uncertainty regarding the future returns on a given portfolio has been measured through the returns on individual assets. The results in Table 10.2 also show that for both approaches the possibilistic measures have a coherent behavior, that is higher returns and higher asymmetry values are associated with higher risk. Note also that portfolio X_4 dominates portfolio X_6 in both cases, the latter thus being inefficient. Moreover, since portfolios X_5 and X_6 have negative skewness they should be rejected if we are looking for portfolios with positive skewness, as will be stated in Sect. 10.4.

For the portfolios analyzed, the width of the support of $\tilde{R}_{P_i}(x)$ is greater than the width of the support of $\tilde{X}_i$, the cores being of similar length. This fact could mean that the fuzzy representation of the portfolio return is more imprecise when it is evaluated through the historical returns on the individual assets. Figure 10.1 shows the fuzzy representation of returns on the portfolios X_1 and X_5. It does not seem easy either to compare both fuzzy representations of uncertainty or to obtain conclusive results for the general statement of fuzzy portfolio selection problems.

In the next section we present certain multi-objective programs for portfolio selection based on the above possibilistic measures of return and risk. Without loss of generality they deal with the approach which directly builds the fuzzy return on a given portfolio, that is $\tilde{X}$.

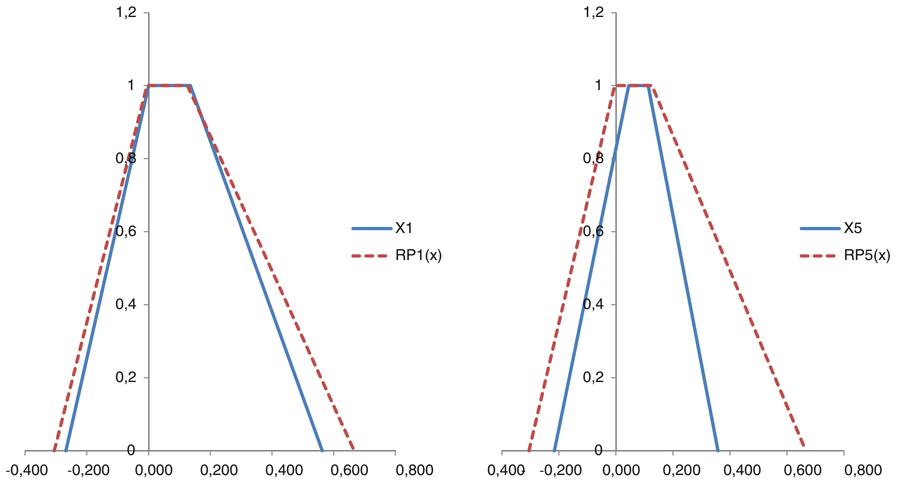


Fig. 10.1 Fuzzy representation of the return on given portfolios. The *left-hand graph* corresponds to portfolio X_1, and the *right-hand graph* to portfolio X_5. The *blue solid lines* correspond to $\tilde{X}_i$ and the *red dashed lines* correspond to $\tilde{R}_{P_i}(x)$, for $i = 1$ and 5, respectively

10.4 Multi-objective Possibilistic Models for Portfolio Selection

Recently, a few multi-criteria decision-making models have been proposed for determining appropriate portfolios in risk-return trade-off assuming a fuzzy representation of the uncertainty regarding future returns [8, 28, 40]. The multi-objective formulation allows consideration of the more realistic situation in which several conflicting goals competed in the allocation decision, providing both flexibility and a large set of choices for the decision maker: the Pareto-optimal set, whose elements are called efficient solutions. On the other hand, from a practical point of view, taking investors' preferences into account implies the inclusion of several types of constraints, which may change the feasible region in which the optimal portfolio must be selected and may also transform the type of optimization problem that must be solved in such a way that it could be considerably more difficult to solve the new problem than the original one. Therefore, the use of suitable mathematical programming techniques is necessary for solving the portfolio selection problem [56].

The mathematical formulation of a multiple criteria optimization problem is as follows:

$$\begin{aligned} (\text{MOP})\ \text{Max}\ & [f_1(x), \ldots, f_r(x)] \\ \text{s.t.}\ \ & x \in S, \end{aligned}$$

where f_i are deterministic functions and $S \in \Re^n$ is the feasible set. This problem also defines the objective feasible region $Z = \{z \in \Re^r : z = f(x),\ x \in S\}$ in the objective space $\Re^r$, r being the number of objectives (see, for instance, [16]).

In the optimization problems with multiple objectives, the set of solutions is composed of all those elements in the decision space S for which the corresponding objective vector cannot be improved in any dimension without another one deteriorating. To characterize the efficient solutions for the MOP problem we use the usual notion of Pareto optimality which determines how one alternative dominates another alternative. Let us recall some useful definitions.

Definition 10.8. Let $z^* \in Z$. Then z^* is a non-dominated solution for MOP if and only if there is no other $z \in Z$ such that $z_i \geq z_i^*$ for $i = 1, \ldots, r$, with strict inequality for at least one of them. Otherwise, $z^* \in Z$ is a dominated solution.

Definition 10.9. A decision vector $x \in S$ is said to be efficient for MOP if and only if $z = f(x)$ is a non-dominated solution.

The set of all efficient points is usually denoted by E and is called the efficient set. The inefficient points are those whose image in Z is a dominated solution. Note that when the goal is to minimize one objective f_i the definitions are analogous, taking into account that $\text{Min} f_i(x) = -\text{Max}(-f_i(x))$.

10.4.1 Possibilistic Mean–Downside Risk–Skewness Model

Here we propose to deal with certain multi-objective decision problems associated with the possibilistic mean–downside risk–skewness model (MDRS), where three objective functions corresponding to the crisp mean values of these possibilistic moments are considered. Note that the objective functions are nonlinear because they depend on the sample percentiles of the returns on the portfolio X. As is usual for selecting efficient portfolios, we propose to maximize the odd moments while minimizing the downside risk value. This multi-objective possibilistic portfolio selection problem can then be formulated as follows:

$$\begin{aligned} (\textbf{MDRS})\ & \text{Max } z_1 = \bar{E}(\tilde{X}) \\ & \text{Min } \ z_2 = \bar{w}(\tilde{X}) \\ & \text{Max } z_3 = S(\tilde{X}) \\ & \text{s.t.} \quad x \in S. \end{aligned} \tag{10.16}$$

Therefore, a non-dominated portfolio must offer the highest level of expected return for a given level of risk and skewness and the lowest level of risk for a given level of return and skewness. On the other hand, an explicit characterization of the decision space S is needed in order to know where the feasible solutions must be found, and then some constraints must be included to incorporate investors' preferences into the model. For instance, concerning an investor's opinion, some of the following constraints should be added:

1. Limits to the budget to be invested in every asset.
   (a) *Lower bounds:* Indicate the minimum level below which an asset is not purchased and also imply an explicit diminution of the fund invested in the sharing portfolio.
   (b) *Upper bounds:* Limit the percentage of the budget in a given asset and it could imply more diversification in the investment.
2. *Portfolio size:* Explicit specification of the number of assets in the portfolio or an explicit rank-size.
3. Limits to the percentage that is invested in one group of assets $J \subset \{1,\dots,n\}$, which are considered the basic unit of investment.
4. Achieving a given level of expected liquidity, and so on.

Note that the above constraints are implicitly related because of the requirement to invest of the total budget and they define different subsets in the n-simplex, requiring the introduction of binary and/or integer variables. Some heuristic algorithms have been proposed to deal with the optimization problems that incorporate these constraints into the mean–variance modeling approach [1, 14, 15].

In what follows we will consider finite upper and lower bounds for the asset weight and cardinality constraints, in such a way that the decision space is mathematically stated as follows:

$$S=\left\{x\in\Re^n : \sum_{j=1}^{n} x_j = 1, l_j \le x_j \le u_j, l_j \ge 0, k_l \le c(X) \le k_u\right\}, \qquad (10.17)$$

where $c(X)$ is the number of positive proportions in portfolio X, that is $c(X) = \text{rank}(\text{diag}(X))$. It is well known that this cardinality constraint involves a quasi-concave function, $c(X)$, which implies that optimization problems with this feasible set are NP-hard [9]. Then, for solving the nonlinear multi-objective decision problem MDRS, we use a meta-heuristic procedure that independently manages the historical information about the returns on the assets and the investors' preferences.

In order to generate efficient portfolios taking into account the goals of the MDRS model, we will apply a multi-objective evolutionary algorithm which has been prepared to deal with two conflicting objective functions. We then deal with two alternative bi-objective optimization problems which incorporate the third goal as a constraint, in the following way:

$$\begin{aligned}(\text{MDRS}_1)\ \text{Min}\ & z_2 = \bar{w}(\tilde{X})\\ \text{Max}\ & z_3 = S(\tilde{X})\\ \text{s.t.}\ & \bar{E}(\tilde{X}) \ge \rho\\ & x \in S,\end{aligned} \qquad (10.18)$$

where ρ is a given expected return, which is usually the rate offered for risk-free investment, and

$$
\begin{array}{lll}
(\mathrm{MDRS}_2) & \mathrm{Max} & z_1 = \bar{E}(\tilde{X}) \\
 & \mathrm{Min} & z_2 = \bar{w}(\tilde{X}) \\
 & \mathrm{s.t.} & S(\tilde{X}) \geq \gamma \\
 & & x \in S.
\end{array} \tag{10.19}
$$

In a previous work we dealt with a bi-objective optimization problem with the same goals as the MDRS$_2$ problem, but without the requirement of positiveness for the coefficient of asymmetry [7]. There we proposed a heuristic procedure for generating the approximate Pareto frontier which is the basis of our multi-objective evolutionary algorithm for solving the MDRS problem.

10.5 An Evolutionary Algorithm for Multi-objective Constrained Fuzzy Portfolio Selection

Evolutionary algorithms (EA) are population-based stochastic heuristic procedures based on the principles of natural selection. Starting with a random initial population, an evolutionary algorithm searches through a solution space by evaluating a set of possible candidates. After the best individuals are selected, new individuals are created for the next generation through random mutation and crossover. Then, the generational cycle is repeated a number of times until convergence.

The more popular evolutionary procedures are genetic algorithms (GA), originally proposed by Holland [26], which have been successfully applied in different fields of decision-making theory. Recently, concerning portfolio optimization some GA-based portfolio selection approaches have been proposed leading with MV and MAD models with additional constraints such as minimum transaction lots, cardinality size, buy-in thresholds, and transactions costs [15, 41, 53]. With respect to the evolutionary approaches for approximating the efficient frontier of multi-objective portfolio selection problems, Anagnostopoulos et al. [1] provide an interesting performance comparison among the more usual multi-objective evolutionary techniques.

10.5.1 Description of the Algorithm

Our proposal for solving the multi-objective optimization problems in (10.18) and (10.19) is to use a multi-objective evolutionary algorithm (MOEA), which has been specifically developed for dealing with admissible portfolios that meet (10.17) and for approximating their corresponding Pareto frontier. Let us describe the basis of our procedure, which follows the general framework outlined by Laumanns et al. [35] and incorporates a dominance method for sorting individuals based on the objective function values (see, for instance, [17]).

We use a standard real-valued vector to represent the proportions of the budget invested in the assets of a given portfolio: X. The procedure works with two populations of individuals. The first population, which is randomly generated in the initialization step, always has the same number of admissible individuals, $x \in S$, while the second one, which maintains the non-dominated solutions, attains different sizes depending on the number of non-dominated solutions in the current generation. At each generation, the quality of an individual is evaluated and the population is sorted according to each objective function values; the procedure rejects those individuals which do not meet the additional constraint. Once the non-dominated individuals have been identified we use them for building the current approximate Pareto frontier: the upper boundary of one generation.

Then, the algorithm selects the best solutions by measuring their distance to the current upper boundary and applies them over a mutation operator which slightly perturbs a pair of randomly selected proportions. Some learning rules are introduced for identifying which are the most interesting assets to being involved in the next generations. The individuals of the offspring population must be admissible portfolios and must contain all the non-dominated portfolios of the previous generation. In order to decide if the convergence has been met, the algorithm measures the distance between two successive upper boundaries, and stops if it is less than a given C. Then, the last approximate Pareto frontier contains the selected portfolios.

10.5.1.1 Experiment Settings

In our experiments we apply a generational genetic population strategy with a population of 500 individuals for each portfolio size, the number of securities in the portfolio varying between $k_l = 6$ and $k_u = 9$. We use an elitism mechanism that selects 20% of the better individuals from the current population. The selection mechanism prefers individuals that are better than other individuals in at least one objective value, i.e. that they are not dominated by another individual. The procedure then maintains the currently approximated Pareto-frontier, including all those individuals in the elite set. We apply a local mutation constant p_0 that takes values in the set $\{0.050, 0.020, 0.010, 0.001\}$. If the stopping criterion is not satisfied, for $C = 10^{-4}$, the algorithm builds 50 generations. These parameters were selected from preliminary experiments (see, for instance, [7]). The algorithm was implemented in R language (http://www.r-project.org/) and runs on a personal computer.

10.5.2 Numerical Results

In this section we report the results that we have obtained on randomly generated portfolios, X, whose risks and returns have been evaluated using a historical data set from the Spanish Stock Exchange in Madrid. We consider the weekly returns on 27 assets from the Spanish IBEX35 index between January 2007 and December 2009. We took the observations of the Wednesday prices as an estimate of the weekly prices. The sample returns r_{tj} for $t = 1, \ldots, 152$ and $j = 1, \ldots, 27$ are

$$r_{tj} = \frac{p_{(t+1)j} - p_{tj}}{p_{tj}}, \tag{10.20}$$

where p_{tj} is the price of asset jth on Wednesday of week tth. For every portfolio X we evaluate its weekly return using (10.12) and we assume a trapezoidal fuzzy representation of the uncertainty associated to its future weekly return. The core and support of $\tilde{X}$ are given by (q_{40}, q_{60}) and (q_5, q_{95}), respectively, q_h being the h percentile of the sample $\{r_t(X)\}$, and then its expected return, downside risk, and coefficient of skewness are obtained.

Let us assume that the diversification parameters are given by $l_j = 0$ and $u_j = 0.2$, for all j, and that the right-hand side values for the corresponding possibilistic moments are fixed as $\rho = 0.001$ and $\gamma = 0.01$ in (10.18) and (10.19), respectively.

In order to analyze the performance of our evolutionary algorithm we have solved numerous instances of the problems defined in (10.18) and (10.19), with different portfolio sizes. The number of positive proportions in portfolio k is set from 6 to 9. Note that because of the upper bound value, the minimum number of assets that compose a portfolio is 5. In addition, we decide to consider up to 9 assets for an admissible portfolio following the suggestion given in [15], which points out that investors should not consider k values above one-third of the total number of assets because of dominance relationships. Each configuration was randomly run 5 times for both multi-objective programs, using the same set of seeds. Let us show some of the results obtained.

10.5.2.1 Possibilistic Downside Risk–Skewness Model

Figure 10.2 shows the downside risk and possibilistic skewness for the first and final generations of 5 independent runs for (10.18), $k = 9$ being the number of assets in every portfolio. The figure clearly shows the generational improvement between these two generations of 2,500 portfolios and the discontinuities which will appear in the efficient frontier because of the cardinality constraint. Note that for the initial population, the algorithm randomly generates portfolios with negative skewness and very diversified risk values. Finally, it converges towards a frontier with an important decrease in risk.

The fact of having discontinuities and missing parts in the Pareto frontier is more clearly shown in Fig. 10.3, where the corresponding efficient frontiers for different portfolio sizes are represented, for k from 6 to 9. Since all of them have been obtained using populations of 2,500 portfolios, the plot shows the efficient frontiers of a set of 10,000 admissible portfolios. Note that the effect of portfolio size can be extremely hard if the investors decide to set this value to 6 assets. However, the dominance relationship is not clearly stated among portfolios of different sizes, at least with respect to risk and skewness.

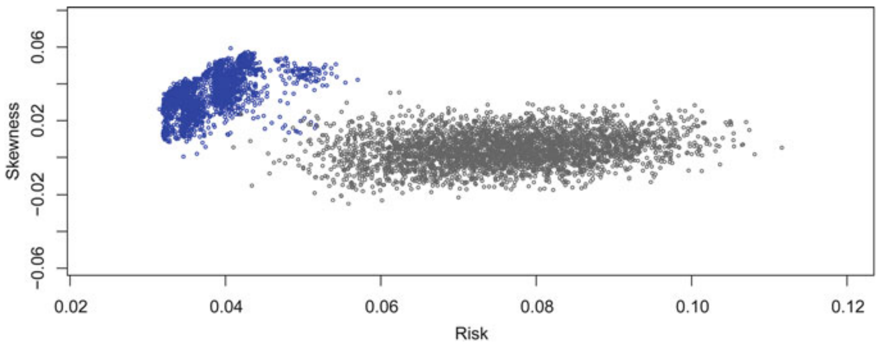


Fig. 10.2 Downside risk and possibilistic coefficient of skewness of the first and final generations obtained for $MDRS_1$ with $k = 9$. The *black points* correspond to the portfolios of the first generation, and the *blue points* to the final generation. Population size: 2,500 portfolios

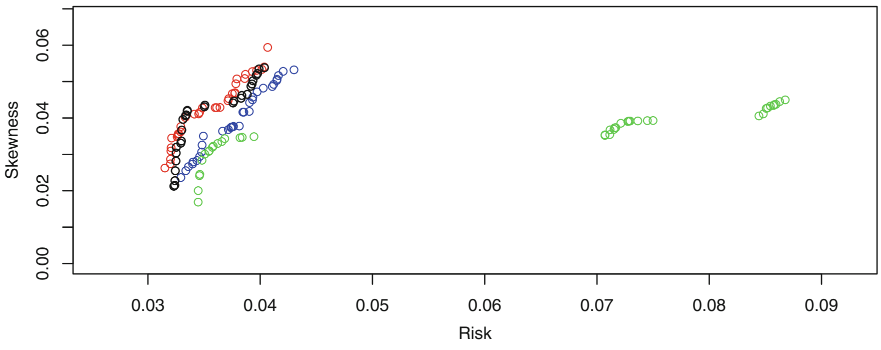


Fig. 10.3 Efficient frontiers of $MDRS_1$ for different portfolio sizes from $k = 6$ to $k = 9$. The *green points* correspond to $k = 6$, the *black points* to $k = 7$, the *blue points* to $k = 8$, and the *red ones* to $k = 9$

We therefore apply our procedure assuming that portfolios may alternatively contain 7, 8, or 9 assets. Figure 10.4 plots the coefficient of skewness and the downside risk corresponding to the 1,500 portfolios of the first and last generations. The final generation is now more explicitly defined, although its efficient frontier also presents a few discontinuities. However, there are not too many differences between the frontiers obtained by dealing with different portfolio sizes (population size: 10,000 portfolios) and this last frontier obtained by randomly building 1,500 portfolios at each generation.

Concerning the expected return of non-dominated portfolios (risk–skewness trade-off) shown in Fig. 10.4, their pairs of risk-return values are shown in Fig. 10.5. Note that some of the portfolios selected by solving $MDRS_1$ are not efficient in the risk-return trade-off.

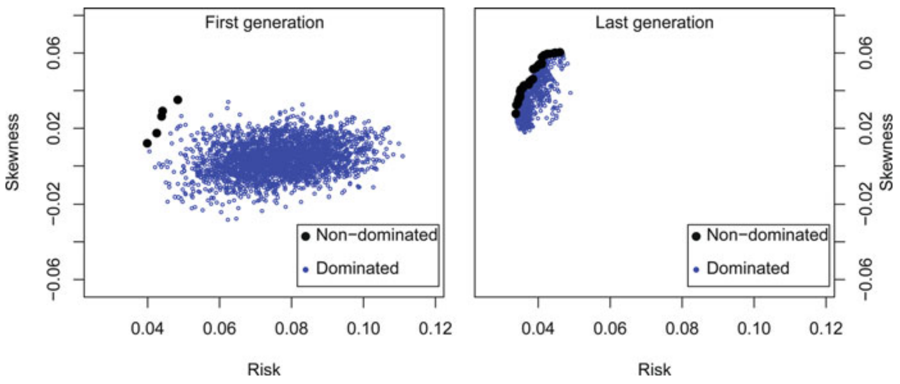


Fig. 10.4 Downside risk and possibilistic coefficient of skewness of the first and final generations obtained by solving $MDRS_1$ for k from 7 to 9. Population size: 1,500 portfolios

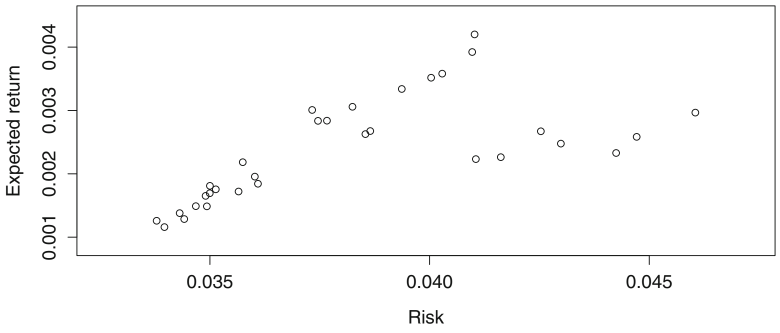


Fig. 10.5 Downside risk and expected return of the portfolios of the Pareto frontier of the final generation obtained by solving $MDRS_1$ for $k \in [7,9]$

On the other hand, since increasing the k value implies increasing diversification, it does not seem that this fact provides any loss of skewness, at least in this optimization framework. In fact, Fig. 10.3 shows that by increasing k we can obtain portfolios with greater values of the skewness coefficient.

10.5.2.2 Possibilistic Mean–Downside Risk Model

Analogously, we have performed the same experiments using the $MDRS_2$ problem and the results are presented in Fig. 10.5 for $k = 7$ and Fig. 10.6 for k from 6 to 9. Again, fixing the number of assets in the portfolio to 6 would not be recommended. In addition, it is also the most time-consuming experiment. Note that the rank of the expected return values has increased considerably with respect to the results obtained by solving $MDRS_1$. Now more risky portfolios with more expected benefits

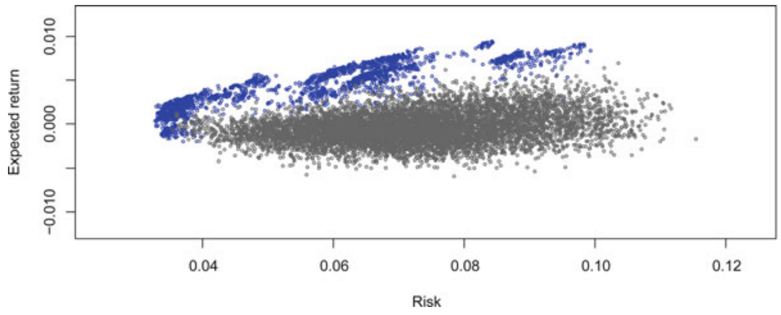


Fig. 10.6 Downside risk and expected return on the portfolios of the first and final generations obtained by solving $MDRS_2$ with $k = 7$. Population size: 2,500 portfolios

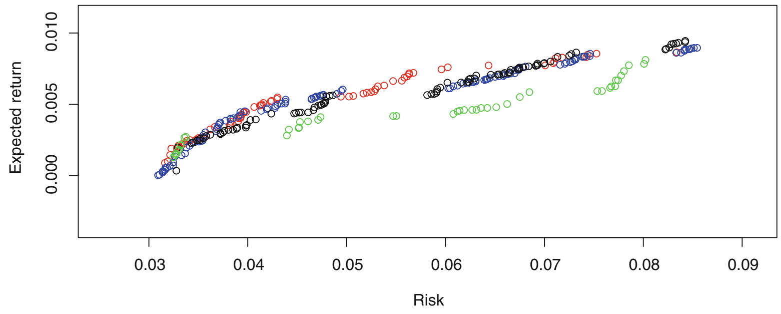


Fig. 10.7 Efficient frontiers of $MDRS_2$ for different portfolio sizes from $k = 6$ to $k = 9$. The *green points* correspond to $k = 6$, the *black points* to $k = 7$, the *blue points* to $k = 8$, and the *red ones* to $k = 9$

are provided for every experiment. However, what is not clearly established are the dominance relationships among the corresponding efficient frontiers where the number of assets in the portfolio is increased (Fig. 10.7).

Figure 10.8 jointly plots the downside risk of and expected return values on the 1,500 portfolios of the first and last generations obtained by applying our evolutionary algorithm to solve $MDRS_2$ for $k \in [7,9]$. Figure 10.9 shows the downside risk and possibilistic coefficient of skewness for the non-dominated portfolios of the last generation. Note that when the risk value increases, the portfolios obtained will be inefficient in the risk–skewness trade-off.

This result reinforces the idea that the inclusion of skewness in the portfolio optimization scheme can play an interesting role for investors with great risk aversion because it can provide them with alternative investment strategies. However, it does not seem to be useful if the investors want to assume bigger risks because in the risk–skewness trade-off the efficient frontier is obtained with lower risk values, which are related to portfolios with lower expected returns.

Concerning the use of our evolutionary algorithm for providing efficient and appropriate portfolios for the investors, the above results show the importance

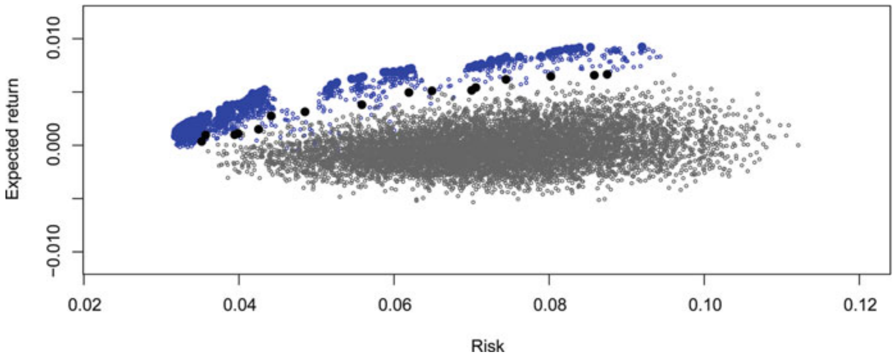


Fig. 10.8 Downside risk and expected return on the portfolios of the first and last generations with $k \in [7,9]$ and $MDRS_2$. The *black points* correspond to the first generation, and the *blue ones* to the last generation. Population size: 1,500 portfolios

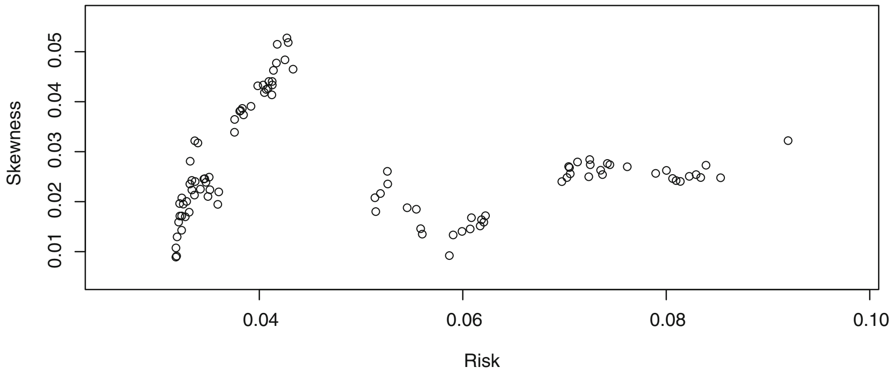


Fig. 10.9 Downside risk and possibilistic skewness of the portfolios of the Pareto frontier of the final generation obtained by solving $MDRS_2$ for $k \in [7,9]$

of suitably identifying their risk profiles. Then, different strategies based on the mean–downside risk–skewness model can be developed in order to obtain different investment proposals with portfolios satisfying the explicit preferences of the investors. This modeling approach can also be useful for providing information concerning the influence of bounds and cardinal sizes over the expected return on and risk of the investment, which could be useful to support her or his decision making.

10.6 Conclusions

Concerning quantification of the uncertainty regarding future returns on risky assets and given portfolios we propose to use LR-fuzzy numbers, whose possibilistic moments measure the risk and profitability of the investment. In order to directly approximate the contemporary relationship among the returns on the assets that compose a portfolio, we propose to consider the returns on a given portfolio as the historical data set instead of considering the individual returns on the assets as the data set.

We extend the mean–downside risk model into a mean–downside risk–skewness model using interval-valued expectations and higher possibilistic moments. We then formulate a new fuzzy portfolio selection problem in which trading requirements and investor preferences are introduced by means of discrete constraints. We propose to solve it by applying multi-objective optimization techniques based on evolutionary searches.

We present some numerical results for the possibilistic mean–downside risk–skewness model for trapezoidal fuzzy numbers. We use two different strategies for finding the efficient frontier and analyze the effect of introducing the possibilistic coefficient of skewness as a goal or as a constraint in the multi-objective optimization problems. Our multi-objective evolutionary algorithm is effective for solving these difficult optimization problems and for providing efficient frontiers that only take into account portfolios that meet investor preferences. Thus, different investment proposals suitably categorized by different risk tendencies can be proposed to the investors.

In our opinion, this multi-objective evolutionary algorithm could be a good strategy for finding suitable portfolios in the approximation Pareto frontier in those situations in which the description of the data set is also made with LR-fuzzy numbers of different shapes, because the analysis of the quality of a portfolio is only based on the specific uncertainty of every portfolio.

Acknowledgements This research was partially supported by the Ministerio de Ciencia e Innovación of Spain under grant number MTM2008-03993.

This manuscript was prepared during a research stay of E. Vercher at the Centre for Interdisciplinary Mathematics (CIM), Uppsala University (Sweden), supported by the Vicerectorat d'Investigació i Política Científica, University of València (Spain).

References

1. K.P. Anagnostopoulos, G. Mamanis, A portfolio optimization model with three objectives and discrete variables. Comp. Oper. Res. **37**, 1285–1297 (2010)
2. M. Arenas, A. Bilbao, M.V. Rodríguez, A fuzzy goal programming approach to portfolio selection. Eur. J. Oper. Res. **133**, 287–297 (2001)
3. E. Ballestero, C. Romero, Portfolio selection: A compromise programming solution. J. Oper. Res. Soc. **47**, 1377–1386 (1996)

4. M. Bazaraa, H. Sherali, C. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd edn. (Wiley, New York, 2006)
5. R. Bellman, L.A. Zadeh, Decision-making in a fuzzy environment. Manag. Sci. **17**, 141–164 (1970)
6. J.D. Bermúdez, J.V. Segura, E. Vercher, A fuzzy ranking strategy for portfolio selection applied to the Spanish stock market, in *Proceedings of the 2007 IEEE International Conference on Fuzzy Systems* (2007), pp. 787–790
7. J.D. Bermúdez, J.V. Segura, E. Vercher, A multi-objective genetic algorithm for cardinality constrained fuzzy portfolio selection. Fuzzy Set. Syst. **188**, 16–26 (2012)
8. R. Bhattacharyya, S. Kar, D.D. Majumber, Fuzzy Mean-Variance-skewness portfolio selection models by interval analysis. Comp. Math. Appl. **61**, 126–137 (2011)
9. S.P. Boyd, L. Vandenberghe, *Convex Optimization* (Cambridge University Press, Cambridge, 2004)
10. W. Briec, K.Kerstens, O. Jokund, Mean-variance-skewness portfolio performance gauging. A general shortage function and dual approach. Manag. Sci. **53**, 135–149 (2007)
11. C. Carlsson, R. Fullér, On possibilistic mean value and variance of fuzzy numbers. Fuzzy Set. Syst. **122**, 315–326 (2001)
12. C. Carlsson, R. Fullér, P. Majlender, A possibilistic approach to selecting portfolios with highest utility score. Fuzzy Set. Syst. **131**, 13–21 (2002)
13. S. Chanas, On the interval approximation of a fuzzy number. Fuzzy Set. Syst. **122**, 353–356 (2001)
14. T.-J. Chang, N. Meade, J.E. Beasley, Y.M. Sharaiha, Heuristics for cardinality constrained portfolio optimization. Comp. Oper. Res. **27**, 1271–1302 (2000)
15. T.-J. Chang, S.-Ch. Yang, K.-J. Chang, Portfolio optimization problems in different risk measures using genetic algorithm. Expert Syst. Appl. **36**, 10529–10537 (2009)
16. V. Chankong, Y.Y. Haimes, *Multiobjective Decision Making: Theory and Methodology* (North Holland, New York, 1983)
17. C.A.C. Coello, Evolutionary multi-objective optimization: A historical view of the field. IEEE Comput. Intell. Mag. **1**(1), 28–36 (2006)
18. M. Delgado, M.A. Vila, W. Voxman, On a canonical representation of fuzzy numbers. Fuzzy Set. Syst. **93**, 125–135 (1998)
19. D. Dubois, H. Prade, The mean value of a fuzzy number. Fuzzy Set. Syst. **24**, 279–300 (1987)
20. D. Dubois, H. Prade, Fuzzy numbers: an overview, in *Analysis of Fuzzy Information*, ed. by J. Bezdek (CRC Press, Boca Raton, 1988), pp. 3–39
21. D. Dubois, H. Prade, *Fundamentals of Fuzzy Sets* (Kluwer, Boston, 2000)
22. Y. Fang, K.K. Lai, S.Y. Wang, Fuzzy portfolio optimization, in *Lecture Notes in Economics and Mathematical Systems*, vol. 609 (Springer, Berlin, 2008)
23. R. Fullér, P. Majlender, On weighted possibilistic mean and variance of fuzzy numbers. Fuzzy Set. Syst. **136**, 363–374 (2003)
24. P. Grzegorzewski, Nearest interval approximation of a fuzzy number. Fuzzy Set. Syst. **130**, 321–330 (2002)
25. T. Hasuike, H. Katagiri, H. Ishii, Portfolio selection problems with random fuzzy variable returns. Fuzzy Set. Syst. **160**, 2579–2596 (2009)
26. J.H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence* (University of Michigan Press, Michigan, 1975)
27. X. Huang, A new perspective for optimal portfolio selection with random fuzzy returns. Inform. Sci. **177**, 5404–5414 (2007)
28. P. Jana, T.K. Roy, S.K. Mazumder, Multi-objective possibilistic model for portfolio selection with transaction cost. J. Comput. Appl. Math. **228**, 188–196 (2009)
29. J.J. Júdice, C.O. Ribeiro, J.P. Santos, A comparative analysis of the Markowitz and Konno portfolio selection models. Investigaão Operacional **23**(2), 211–224 (2003)
30. H. Konno, K. Suzuki, A mean-variance-skewness optimization model. J. Oper. Res. Soc. Jpn. **38**, 137–187 (1995)

31. H. Konno, H. Yamazaki, Mean-absolute deviation portfolio optimization model and its application to Tokyo stock market. Manag. Sci. **37**, 519–531 (1991)
32. H. Konno, H. Waki, A. Yuuki, Portfolio optimization under lower partial risk measures. Asia Pac. Financ. Market **9**, 127–140 (2002)
33. V. Lacagnina, A. Pecorella, A stochastic soft constraints fuzzy model for a portfolio selection problem. Fuzzy Set. Syst. **157**, 1317–1327 (2006)
34. T. Lai, Portfolio selection with skewness: A multiple-objective approach. Rev. Quant. Finance Account. **1**, 293–305 (1991)
35. M. Laumanns, E. Zitzler, L. Thiele, A unified model for multi-objective evolutionary algorithms with elitism, in *Proceedings of the 2000 Congress on Evolutionary Computation* (IEEE Press, Piscataway, 2000), pp. 46–53
36. T. León, E. Vercher, Solving a class of fuzzy linear programs by using semi-infinite programming techniques. Fuzzy Set. Syst. **146**, 235–252 (2004)
37. T. León, V. Liern, E. Vercher, Viability of infeasible portfolio selection problems: A fuzzy approach. Eur. J. Oper. Res. **139**, 178–189 (2002)
38. T. León, V. Liern, P. Marco, J.V. Segura, E. Vercher, A downside risk approach for the portfolio selection problem with fuzzy returns. Fuzzy Econ. Rev. **9**, 61–77 (2004)
39. H. Levy, H.M. Markowitz, Approximating expected utility by a function of mean and variance. Am. Econ. Rev. **69**, 308–317 (1975)
40. X. Li, Z, Qin, S. Kar, Mean-Variance-skewness model for portfolio selection with fuzzy returns. Eur. J. Oper. Res. **202**, 239–247 (2010)
41. C.C. Lin, T.Y. Liu, Genetic algorithms for portfolio selection problems with minimum transaction lots. Eur. J. Oper. Res. **185**, 393–404 (2007)
42. J. Lintner, The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets. Rev. Econ. Stat. **47**, 13–37 (1965)
43. D. Maringer, H. Kellerer, Optimization of cardinality constrained portfolios with a hybrid local search algorithm. OR Spectrum **25**(4), 481–495 (2003)
44. H.M. Markowitz, Portfolio selection. J. Finance **7**, 77–91 (1952)
45. H.M. Markowitz, *Portfolio Selection: Efficient Diversification of Investments* (Wiley, New York, 1959)
46. F.J. Ortí, J. Sáez, A. Terceño, On the treatment of uncertainty in portfolio selection. Fuzzy Econ. Rev. **7**, 59–80 (2002)
47. C. Papahristodoulou, E. Dotzauer, Optimal portfolios using linear programming models. J. Oper. Res. Soc. **55**, 1169–1177 (2004)
48. S. Ramaswamy, Portfolio Selection Using Fuzzy Decision Theory. BIS Working Paper no. 59, Bank for International Settlements (1998)
49. A. Saedifar, E. Pasha, The possibilistic moments of fuzzy numbers and their applications. J. Comput. Appl. Math. **223**, 1028–1042 (2009)
50. W.F. Sharpe, Capital asset prices: A theory of market equilibrium under conditions of risk. J. Finance **19**, 425–442 (1964)
51. Y. Simaan, Estimation risk in portfolio selection: The mean variance model versus the mean absolute deviation model. Manag. Sci. **43**, 1437–1446 (1997)
52. R. Slowinski, *Fuzzy Sets in Decision Analysis, Operations Research and Statistics* (Kluwer, Boston, 1998)
53. H. Soleimani, H.R. Golmakani, M.H. Salimi, Markowitz-based portfolio selection with minimum transaction lots, cardinality constraints and regarding sector capitalization using genetic algorithm. Expert Syst. Appl. **36**, 5058–5063 (2009)
54. F. Sortino, R. van der Meer, Downside risk. J. Portfolio Manag. **17**, 27–32 (1991)
55. M.G. Speranza, Linear programming model for portfolio optimization. Finance **14**, 107–123 (1993)
56. R.E. Steuer, Y. Qi, M. Hirschberger, Suitable-portfolio investors, nondominated frontier sensitivity, and the effect of multiple objectives on standard portfolio selection. Ann. Oper. Res. **152**, 297–317 (2007)

57. M. Tamiz, Multi-objective programming and goal programming, in *Lecture Notes in Economics and Mathematical Systems*, vol. 432 (Springer, Berlin, 1996)
58. T. Tanaka, P. Guo, Possibilistic data analysis and its application to portfolio selection problems. Fuzzy Econ. Rev. **2**, 2–23 (1999)
59. E. Vercher, Portfolios with fuzzy returns: Selection strategies based on semi-infinite programming. J. Comput. Appl. Math. **217**, 381–393 (2008)
60. E. Vercher, J.D. Bermúdez, A possibilistic mean-downside risk-skewness model for efficient portfolio selection. Technical Report: TR Departament Estadística i Investigació Operativa, Universitat de València (2011)
61. E. Vercher, J.D. Bermúdez, J.V. Segura, Fuzzy portfolio optimization under downside risk measures. Fuzzy Set. Syst. **158**, 769–782 (2007)
62. S.Y. Wang, Y.S. Xia, Portfolio selection and asset pricing, in *Lecture Notes in Economics and Mathematical Systems*, vol. 514 (Springer, Berlin, 2002)
63. J. Watada, Fuzzy portfolio selection and its applications to decision making. Tatra Mountains Mathematical Publication **13**, 219–248 (1997)
64. P. Xidonas, G. Mavrotas, C. Zopounidis, J. Psarras, IPSSIS: An integrated multicriteria decision support system for equity portfolio construction and selection. Eur. J. Oper. Res. **210**, 398–409 (2010)
65. L.A. Zadeh, Fuzzy sets. Inf. Contr. **8**, 338–353 (1965)
66. L.A. Zadeh, Fuzzy sets as a basis for a theory of possibility. Fuzzy Set. Syst. **1**, 3–28 (1978)
67. H.J. Zimmermann, Fuzzy Programming and linear programming with several objective functions. Fuzzy Set. Syst. **1**, 45–55 (1978)
68. H.J. Zimmermann, *Fuzzy Set Theory and its Applications*, 4th edn. (Kluwer, Boston, 2001)
69. C. Zopounidis, M. Doumpos, Multi-criteria decision aid in financial decision making: Methodologies and literature review. J. Multi-Criteria Decis. Anal. **11**, 167–186 (2002)

Chapter 11
Financial Evaluation of Life Insurance Policies in High Performance Computing Environments

Stefania Corsaro, Pasquale Luigi De Angelis, Zelda Marino, and Paolo Zanetti

Abstract The European Directive Solvency II has increased the request of stochastic asset–liability management models for insurance undertakings. The Directive has established that insurance undertakings can develop their own "internal models" for the evaluation of values and risks in the contracts. In this chapter, we give an overview on some computational issues related to internal models. The analysis is carried out on "Italian style" *profit-sharing life insurance policies* (PS policy) with minimum guaranteed return. We describe some approaches for the development of accurate and efficient algorithms for their simulation. In particular, we discuss the development of parallel software procedures. Main computational kernels arising in models employed in this framework are stochastic differential equations (SDEs) and high-dimensional integrals. We show how one can develop accurate and efficient procedures for PS policies simulation applying different numerical methods for SDEs and techniques for accelerating Monte Carlo simulations for the evaluation of the integrals. Moreover, we show that the choice of an appropriate probability measure provides a significative gain in terms of accuracy.

11.1 Introduction

The European Directive 2009/138 (Solvency II) [19] has more and more increased the request of stochastic asset–liability management (ALM) models for insurance undertakings. The article 44 of the Directive states that "insurance and reinsurance undertakings shall have in place an effective risk-management system—RSM—

S. Corsaro (✉) • P.L. De Angelis • Z. Marino • P. Zanetti
Dipartimento di Statistica e Matematica per la Ricerca Economica, Università degli Studi di Napoli "Parthenope", via Medina 40, 80133 Napoli, Italy
e-mail: corsaro@uniparthenope.it; deangelis@uniparthenope.it; marino@uniparthenope.it; zanetti@uniparthenope.it

comprising strategies, processes and reporting procedures necessary to identify, measure, monitor, manage and report, on a continuous basis the risks, at an individual and at an aggregated level, to which they are or could be exposed, and their interdependencies."

The Solvency II Directive has established that insurance undertakings can develop their own "internal models" for the evaluation of relevant quantities, both values and risks, in the contracts. The "internal model" is defined by the International Association of Insurance Supervisors as "a risk management system developed by an insurer to analyse its overall risk position, to quantify risks and to determine the economic capital required to meet those risks" [27]. Nevertheless, in order to make the internal models of "effective" use, it is fundamental to obtain responses in a suitable turnaround time; therefore, the computational performance of the evaluation process plays a crucial role in this framework.

In this chapter, we focus on some computational issues related to internal models; for the internal model validation it is a challenging matter to focus on their numerical solution, with the aim of obtaining adaptive solution processes, that is, capable of being properly scaled in order to balance accuracy and computational efficiency on demand, depending on the evaluation context.

The analysis is carried out on "Italian style" *profit-sharing life insurance policies* (PS policies) with minimum guarantees. In these contracts, the benefits which are credited to the policyholder are indexed to the annual return of a specified investment portfolio, called the *segregated fund* (in Italian *gestione separata*). In Italian insurance market, the crediting mechanism typically guarantees a minimum to the policyholder. A profit-sharing policy is a "complex" structured contract, with underlying the segregated fund return; the models for values and risk evaluation of the policy must provide "market-consistent valuation," thus requiring the use of a stochastic framework and of Monte Carlo (MC) simulation techniques.

The quantities in a single contract, defined in condition of financial and actuarial uncertainty, have to be computed with great accuracy and in an "adequate" time in order to effectively support undertakings in coming to decisions and to allow them to quickly act in an appropriate way. The complexity of the contracts and the great number of contracts in the portfolio lead to a complex overall valuation process, thus requiring both accurate and efficient numerical algorithms and *High Performance Computing* (HPC) methodologies and resources.

The literature on the development of high performance procedures for the ALM of PS policy simulation is very poor; a relevant contribution is given in [8] where appropriate technical and technological solutions based on a parallel distributed processing framework (grid) are shown.

In this chapter, we report some results of our research activity on the numerical simulation of profit-sharing policies.

In a first stage of our activity, we focused on the typical computational kernels arising from stochastic models employed for PS policy modeling, aiming at developing accurate numerical algorithms.

Main computational kernels involved in the simulations of PS policies are multidimensional integrals and stochastic differential equations (SDEs). The integrals represent expected values, the SDEs model the diffusion processes describing the time evolution of risk sources. Preliminary experiments were carried out on a single insurance contract; the analysis that we performed is presented in [11].

We then developed a parallel software based on the best-performing numerical schemes. The first HPC environment we considered is a cluster-based system with supercomputing capability assembled with commodity-off-the-shelf (COTS) hardware components and equipped with freely available, standard software. The choice was due to the observation that the use of cluster of personal computers and workstations as platforms for running high performance applications in financial undertakings widely increased, mainly due to their cost-effective nature. This research activity is described in [14].

In the second stage of our work, we have considered the numerical simulation of real ALM portfolios. In particular, we used the dynamic investment strategy with accounting rules (DISAR) ALM system [8] as valuation framework.

In this framework, we analyzed a change of *numéraire* in the stochastic processes for risk sources, since the flexibility of this approach can be particularly valuable in a model with stochastic interest rates. In particular, we analysed the use of the numéraire which defines the *forward risk-neutral measure*. Pricing under the forward measure can provide considerable gains in accuracy, since it allows to discount at a deterministic price deflator, even though the short rate is stochastic. We proposed parallel algorithms for ALM of PS policy portfolios, under both risk-neutral and forward risk-neutral measure, based on the parallelization of Monte Carlo method [12]. The high performance computing environment used to implement and test the algorithms is a Bladecenter architecture consisting of six multi-core Blade; multi-core clusters are more and more popular, with almost all of the Top 500 systems (http://www.top500.org) containing processors with a multi-core architecture.

We moreover investigated the use of stochastic models to measure *default risk* in PS policies. The motivation for this has been again the European Directive Solvency II: indeed, it is established that "insurance and reinsurance undertakings may take full account of the effect of risk-mitigation techniques in their internal model, as long as credit risk ..." (art. 121), the default risk being a fundamental component of credit risk. The carried out activity is described in [13].

The rest of this work is organized as follows: in Sect. 11.2 we present the general valuation framework we refer to; in Sect. 11.3 we describe main computational kernels in stochastic models employed for PS policy valuation; in Sect. 11.4 the development of accurate and efficient procedures for PS policy simulation is discussed and in Sect. 11.5 default-risk modeling is considered. Finally, in Sect. 11.6 we give some conclusions.

11.2 Valuation Framework

In this section we describe the main features of the mathematical formalization of the Italian contractual standard for profit-sharing policies. We refer to notation defined in [8, 17, 18]. For an exhaustive analysis of the basic principles and methodological approach for a valuation system of profit-sharing policies with minimum guarantees we address to [9].

In order to describe the profit-sharing mechanism, we consider a single premium pure endowment insurance contract, written at time 0 for a life with age x, with term T years and initial sum insured Y_0. The crediting mechanism generally includes minimum guarantees. Let Y_0 be the initial sum insured and T the term, in years, of the policy. Following a typical interest crediting mechanism, the benefits are readjusted at the end of the year t according to the *revaluation rule*

$$Y_t = Y_{t-1}(1+\rho_t), \qquad t = 1,\dots,T, \tag{11.1}$$

where ρ_t is the so-called *readjustment rate* defined as

$$\rho_t = \frac{\max\{\min\{\psi I_t; I_t - \eta\} - i; \delta^c\}}{1+i}. \tag{11.2}$$

$\psi \in (0,1]$ is the participation coefficient, I_t is the rate of return of the segregated fund in the year $[t-1,t]$, η is the minimum annual rate retained by the insurance undertaking, i is the *technical rate*, and δ^c is the minimum guaranteed annual *cliquet rate*.

All quantities, with obviously the exception of I_t, are contractually defined. The final benefit is given by the insured sum Y_0, raised at the financial readjustment factor Φ_T

$$Y_T = Y_0 \Phi_T, \tag{11.3}$$

where, from (11.1) to (11.3), it follows that

$$\Phi_T = \prod_{k=1}^{T}(1+\rho_k). \tag{11.4}$$

If we denote by $\varepsilon(x,T)$ the event "the aged x insured is alive at time T," then the benefit for the policyholder—the liability of the company—in T is given by

$$Y_T = Y_0 \Phi_T I_{\varepsilon(x,T)},$$

where $I_{\varepsilon(x,T)}$ is the indicator function of $\varepsilon(x,T)$, defining actuarial uncertainty.

In a market consistent valuation framework of the policy the value of the benefits at time $t=0$, $V(0,Y_T)$, can be expressed as the expected value of the payoff at time $t=T$, weighted by a suitable state-price deflator. Both actuarial and financial uncertainty have to be taken into account, anyway these risk sources can be supposed to be mutually independent, thus the expectation can be factorized. Finally, the expectations can be rewritten with a change of measure, employing the so-called risk-neutral probability measure Q. It is well known that any positive martingale, with initial value one, defines a change of probability measure; thus, through the Radon–Nikodym derivative one can switch between suitable probability measures. This process is usually referred to as a change of *numéraire* [4,24], where the numéraire is a non-dividend paying asset with respect to which a probability measure is defined. The numéraire corresponding to the risk-neutral measure is, at time t, the money market account

$$\beta(t,T) = \mathrm{e}^{\int_t^T r(u)\,\mathrm{d}u}.$$

Under Q prices measured in units of the value of the money market account are martingales. Accordingly,

$$V(0,Y_T) = \mathbf{E}^Q\left[\frac{Y_T}{\mathrm{e}^{\int_0^T r(u)\mathrm{d}u}}\middle|\mathscr{F}_0\right]{}_T p_x = \overline{Y}_T\,\mathbf{E}^Q\left[\prod_{k=1}^{T}(1+\rho_k)\,\mathrm{e}^{-\int_0^T r(u)\,\mathrm{d}u}\middle|\mathscr{F}_0\right], \quad (11.5)$$

where $r(t)$ is the instantaneous intensity of interest rate determining the value of the money market account, $\mathscr{F}_t$ is the filtration containing the information about financial events up to time t,[1] ${}_T p_x$ is the risk-neutral probability that an individual aged x will persist for T more years (lapse included), and $\overline{Y}_T = Y_0\,{}_T p_x$ is the *actuarially expected* benefit. Notice that both ρ_t and $r(t)$ are $\mathscr{F}_t$-adapted random variables.

The quantity $V(0,Y_T)$ can be expressed using either a put or a call decomposition (see [8], eq. 4, and [18], p. 91)

$$V(0,Y_T) = B_0 + P_0 = G_0 + C_0, \quad (11.6)$$

where B_0 is the value of a risky investment (base component) and P_0 that of a put option; G_0 is the value of a guaranteed investment and C_0 that of a call option, or—in the jargon of the Directive Solvency II—the policy guaranteed benefits and its future discretionary benefits. These decompositions show that the annual minimum guarantees imply that financial options are embedded in these policies. The "underlying" is the annual return of the reference fund, which in turn is influenced by the "management actions" of the insurer.

[1]In the following formulas, for sake of brevity, it is understood that all the expected values are conditional expectations.

11.3 Stochastic Processes for Risk and Computational Kernels

This section is devoted to the description of the main computational kernels in stochastic models employed for PS policy valuation. In the general valuation framework described in Sect. 11.2, the main computational kernels are multidimensional integrals, representing expected values, and SDEs for risk modeling.

The segregated fund is typically composed of stocks and bonds; thus, risk sources related to them both have to be taken into account. We now consider a typical market model with three sources of uncertainty: interest rate, stock market, and consumer price index (CPI). For each one we present a model mostly used in financial modeling. We will introduce default-risk modeling later in the chapter.

Risk sources, time evolution is modeled by SDEs under a certain probability measure. We introduce the models for risks in a typical risk-neutral setting. Risk adjusted parameters are the only ones required for pricing purpose.

We denote by

$$\tilde{\mathbf{W}} = (\tilde{W}_r, \tilde{W}_p, \tilde{W}_S),$$

the vector containing the risk-neutral Girsanov transformations of a three-dimensional standard Brownian motion driving the time evolution processes of the short rate, the CPI, and the stock market, respectively.

For the interest rate risk, we refer to the well-known one-factor Cox–Ingersoll–Ross (CIR) model [15].

This model has some features which turn to be useful in insurance policy modeling; for instance, it produces nonnegative values for the nominal interest rates and, as reported below, it allows one to obtain closed form for the price of the unitary default-free zero-coupon bond (zcb). On the other hand, in particular situations the CIR model is not able to accurately fit market data; in these cases, other models could be employed, such as the CIR++ [4] and two-factor model proposed by Hull and White [26].

The market short rate at time t, $r(t)$, is assumed to follow the square-root mean-reverting diffusion process

$$\mathrm{d}r(t) = \tilde{\alpha}[\tilde{\gamma} - r(t)]\,\mathrm{d}t + \sigma_r\sqrt{r(t)}\,\mathrm{d}\tilde{W}_r(t), \tag{11.7}$$

with $2\tilde{\alpha}\tilde{\gamma} > \sigma_r^2$. This condition ensures the positivity of the process (11.7). $\tilde{\alpha}$ and $\tilde{\gamma}$ are a linear transformation of the corresponding actual parameters. In this modeling context, a closed form for the price $B(t,T)$ of the unitary default-free zcb with maturity T is available

$$B(t,T) = A_r(t,T)\mathrm{e}^{-\beta_r(t,T)r(t)}, \tag{11.8}$$

where

$$A_r(t,T) = \left[\frac{d_r e^{\phi_r(T-t)}}{\phi_r(e^{d_r(T-t)} - 1) + d_r} \right]^{\nu_r}, \tag{11.9}$$

$$\beta_r(t,T) = \frac{e^{d_r(T-t)} - 1}{\phi_r(e^{d_r(T-t)} - 1) + d_r} \tag{11.10}$$

and

$$d_r = \sqrt{\tilde{\alpha}^2 + 2\sigma_r^2}, \quad \nu_r = \frac{2\tilde{\alpha}\tilde{\gamma}}{\sigma_r^2}, \quad \phi_r = \frac{\tilde{\alpha} + d_r}{2}.$$

In (11.9) and (11.10) the risk-adjusted Brown–Dybvig parametrization [5] is employed.

Since $\log B(t,T)$ is an affine transformation of $r(t)$, the CIR process (11.7) belongs to the so-called *affine class* of interest rate models [24].

For stock market risk we consider the Black and Scholes [2] log-normal process

$$\frac{dS(t)}{S(t)} = (r(t) - d)dt + \sigma_S d\tilde{W}_S(t), \tag{11.11}$$

where d is the dividend yield.

Finally, CPI is described via a log-normal process as well; in particular, we suppose that it evolves as [10]

$$\frac{dp(t)}{p(t)} = \tilde{y}_t dt + \sigma_p d\tilde{W}_p(t), \tag{11.12}$$

where

$$\tilde{y}_t = y_t - \sigma_p$$

is the risk-neutral counterpart of the expected value of inflation at time t

$$y_t = y_\infty + (y_0 - y_\infty)e^{-\alpha_y t}.$$

y_0, y_∞ are the levels of current and long-period expected inflation, respectively; expected inflation is therefore supposed to evolve deterministically with respect to time. The parameter α_y corresponds to the speed of adjustment, that is, y_t is "pulled" towards y_∞ at a speed controlled by α_y.

The short rate and the stock processes are often supposed to be not correlated to the CPI. This is a restrictive hypothesis, but it supplies a simplified mathematical model [10]. We follow this approach, that is, we only model correlation between the short rate and the stock market processes. In particular, we consider a constant correlation between r and S. Let $\mathbf{L}$ be the Cholesky factor of the correlation matrix;

from the mentioned hypotheses it follows that the matrix $\mathbf{L}$ has the following sparsity pattern:

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_{13} & 0 & l_{33} \end{pmatrix},$$

where l_{13} is the fixed correlation level between r and S and $l_{33} = \sqrt{1 - l_{13}^2}$. Let furthermore $\bar{\mathbf{W}} = (\bar{W}_r, \bar{W}_p, \bar{W}_S)$ be the correlated Brownian motion; the following relation holds:

$$\mathrm{d}\bar{\mathbf{W}} = \mathbf{L} \cdot \mathrm{d}\tilde{\mathbf{W}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_{13} & 0 & l_{33} \end{pmatrix} \cdot \begin{pmatrix} \mathrm{d}\tilde{W}_r(t) \\ \mathrm{d}\tilde{W}_p(t) \\ \mathrm{d}\tilde{W}_S(t) \end{pmatrix} = \begin{pmatrix} \mathrm{d}\tilde{W}_r(t) \\ \mathrm{d}\tilde{W}_p(t) \\ l_{13}\mathrm{d}\tilde{W}_r(t) + l_{33}\mathrm{d}\tilde{W}_S(t) \end{pmatrix}, \quad (11.13)$$

thus, when introducing correlation among risk processes, only (11.11) has to be modified according to (11.13).

The integrals involved in PS policy simulation are high-dimensional ones [7,39]; to derive such a representation, let us consider the revaluation factor in (11.4). The function Φ_T depends on the risk sources, expressed, as seen, by means of SDEs that are simulated on a time grid

$$0 = t_0 < t_1 < \cdots < t_n = T$$

therefore, n independent normally distributed random numbers are needed for each process. Let

$$\xi_{\mathbf{i}} = (\xi_{i,1}, \ldots, \xi_{i,n}) \sim N(0,1), \quad i = 1, \ldots, k$$

be the random vector generated to simulate the ith risk source, where k is the number of risk sources; to simulate all the stochastic processes we have to generate the random vector

$$\mathbf{Z} = (Z_1, \ldots, Z_{kn}) = (\xi_{1,1}, \ldots, \xi_{1,n}, \ldots, \xi_{k,1}, \ldots, \xi_{k,n}),$$

then, Φ_T can be regarded as a deterministic function $\Phi_T(\mathbf{Z})$ of the normally distributed vector $\mathbf{Z} \in \Re^{kn}$. As a consequence, for instance, the expected value in (11.5) can be represented by the kn-dimensional integral

$$\mathbf{E}^Q\left[\Phi_T \mathrm{e}^{-\int_0^T r(u)\,\mathrm{d}u}\right] = V(0, \Phi_T) = \int_{\Re^{kn}} \Phi_T(\mathbf{Z}) \frac{\mathrm{e}^{-\mathbf{Z}^T\mathbf{Z}/2}}{(2\pi)^n} \mathrm{d}\mathbf{Z}. \quad (11.14)$$

Typical values for the dimension kn are of order of hundreds, depending on the time horizon and the discretization step.

The integral (11.14) can be transformed into an integral over the kn-dimensional unit cube by means of the substitution $Z_i = G^{-1}(x_i)$ for $i = 1, \ldots, kn$, where G^{-1} denotes the inverse cumulative normal distribution function, thus obtaining

$$V(0, \Phi_T) = I(f) = \int_{[0,1]^d} f(\mathbf{x}) \mathrm{d}\mathbf{x}, \tag{11.15}$$

with $d = kn$ and $f(\mathbf{x}) = \Phi(G^{-1}(\mathbf{x}))$.

11.4 Improving Accuracy and Efficiency

In this section we describe some approaches for the development of accurate and efficient procedures for PS policy simulation.

This purpose can be obviously met in different ways. For instance, one can focus on the solution of computational kernels, investigating, on one hand, various numerical methods for SDEs, on the other hand, techniques for accelerating numerical simulations performed to evaluate multidimensional integrals. This approach turns out to be effective when dealing with "small" portfolios. The simulation of real "large" ALM portfolios requires execution times of order of hours, thus, the gain which one can obtain by most efficient numerical methods is not enough: a huge number of variables are involved and a huge number of conditions have to be taken into account for accurate forecasts in this case. In order to develop software procedures which can provide reliable estimates in a useful turn around time, it is necessary, for example, to act on the mathematical model. We here present a change of *numéraire*, that is, a change of the probability measure. Moreover, one can use HPC methodologies and resources.

As already pointed out, we choose to organize this part of the chapter in accordance with the time stages of our research activity on the topic. We start by focusing on the main kernels in models for PS policy evaluation, thus, we act on the numerical solution of SDEs and multidimensional integrals. In Sect. 11.4.1 we indeed explore the use of different methods for the solution of computational kernels and compare them in terms of performance (accuracy and efficiency).

In the second part, we discuss the simulation of real ALM portfolios of PS policies. In Sect. 11.4.2 we describe a general procedure for this problem and analyze the use of the *forward risk-neutral measure* for improving the accuracy, then reducing the execution time. In Sect. 11.4.3 we describe a parallelization strategy to speed up ALM portfolio simulation procedures.

11.4.1 Improving Accuracy and Efficiency: A First Approach

In Sect. 11.4.1.1 we present some among the mostly used numerical methods for SDEs, selected for having certain desiderable properties. In Sect. 11.4.1.2 we consider integrals; it is well known that high-dimensional integrals are usually solved via Monte Carlo method. We describe the method together with the Antithetic Variates (AV), a variance reduction technique used to improve the convergence rate of MC. In Sect. 11.4.1.3 we support our discussion with some numerical experiments.

11.4.1.1 Investigating Numerical Methods: Solution of SDEs

In this section we analyze some numerical schemes for the solution of the linear SDEs (11.7), (11.11), and (11.12) and compare their performance in terms of accuracy and efficiency in the evaluation of PS policies, with the aim of selecting the most effective ones.

In the numerical solution of SDEs, convergence and numerical stability properties of the schemes play a fundamental role as well as in a deterministic framework. Regarding the stability of a numerical method for SDEs, an important role is played by its region of *absolute stability*, as discussed in [30], since it defines possible restrictions on the maximum allowed step size, ensuring that errors will not propagate in successive iterations. Explicit methods are usually simpler to implement, but implicit methods generally reveal larger stability regions, therefore the bounds imposed on the values of step size are less stringent than for explicit ones. Indeed, both explicit and implicit methods have been investigated in literature; in particular, implicitness has been introduced either in the deterministic part of the equations only, leading to the so-called *drift-implicit methods*, or in both the deterministic and stochastic terms, in *stochastically* or *fully implicit* methods [37].

Several criteria are defined to estimate the approximation error; we essentially distinguish strong criteria from weak ones: strong criteria are based on the pathwise proximity between the continuous random process and the approximated one, weak ones require their probability distribution functions to be closed. When dealing with pricing problems, weak convergence criteria are actually more relevant, since prices are expressed as expected values.

In the following, we briefly describe the methods that we have tested. We selected an explicit method, namely, the well-known Euler method, a drift-implicit one and two fully implicit schemes having different convergence orders. We do not go into details in the description of the methods; we address to the copious existing literature on the topic, in particular to [30] and references therein. We just focus on equation (11.7), since the extension to the others is straightforward.

Let $[0,T]$ be a time interval; we consider a time grid with fixed time step $h > 0$, that is, $t_i = ih$, $i = 0,\ldots,n$. Let us furthermore denote by $\bar{r}$ a time-discrete approximation of function r in (11.7) on the mentioned time grid. In the following, we will sometimes use the notation

$$\bar{r}_i := \bar{r}(ih) = \bar{r}(t_i).$$

The first method we discuss is the well-known explicit Euler stochastic scheme. The explicit Euler approximation of (11.7) is defined by

$$\bar{r}_{i+1} = \bar{r}_i + \tilde{\alpha}[\tilde{\gamma} - \bar{r}_i]h + \sigma_r \sqrt{h\bar{r}_i}\mathrm{d}\bar{W}_{r,i+1} \tag{11.16}$$

with $\bar{r}(0) = r(0)$ and $\mathrm{d}\tilde{W}_{r,1}, \mathrm{d}\tilde{W}_{r,2}, \ldots$ independent, standard normal random variables.

The explicit Euler scheme achieves order-1 weak convergence if appropriate hypotheses on the coefficients of the equation, reported in [24, 30], are satisfied.

As already pointed out, since implicit schemes can reveal better stability properties, we take into account the drift-implicit Euler scheme too [30]; we refer to this as "Implicit Euler" in the following for brevity. This scheme is obtained by making implicit just the pure deterministic term of the equation, while at each time step, the coefficients of the random part of the equation are retained from the previous step. Using the same notations as in (11.16), we have, at each time step,

$$\bar{r}_{i+1} = \bar{r}_i + \tilde{\alpha}[\tilde{\gamma} - \bar{r}_{i+1}]h + \sigma_r \sqrt{h\bar{r}_i}\mathrm{d}\bar{W}_{r,i+1}.$$

Implicit Euler has the same weak order of convergence of corresponding explicit Euler scheme, but the step size can be chosen arbitrarily large. From the computational point of view, it is not more expensive than (11.16), but it can allow better accuracy.

It is well known that Euler schemes for CIR process can lead to negative values since the Gaussian increment is not bounded from below. Different fixing techniques are applied to get positive solutions. Negative values can be either reflected about zero (*reflection*) or set to zero (*absorption*). Moreover, partial correction to equations has been considered, as in [16, 25]. See [33] for a detailed discussion. In our simulations, we apply absorption technique.

Milstein [36] proposed a order-2 weak convergence method. We here describe a simplified version of this method for practical implementation as shown in [24] and in [30], approximating the diffusion process (11.7) by the following expansion:

$$\begin{aligned}\bar{r}_{i+1} &= \bar{r}_i + ah + b\sqrt{h}\mathrm{d}\tilde{W}_{r,i+1} + \frac{1}{2}\left(a'b + ab' + \frac{1}{2}b^2 b''\right) h\sqrt{h}\mathrm{d}\bar{W}_{r,i+1} \\ &\quad + \frac{1}{2}bb'h\left[\mathrm{d}\bar{W}_{r,i+1}^2 - 1\right] + \left(aa' + \frac{1}{2}b^2 a''\right)\frac{1}{2}h^2,\end{aligned}$$

where $a = \tilde{\alpha}\,[\tilde{\gamma} - r(t)]$, $b = \sigma_r\sqrt{r(t)}$, with a, b and their derivatives all evaluated at time t_i. This scheme is more accurate than the Euler method, but it is computationally more expensive.

We finally tested a fully implicit, positivity-preserving Euler scheme introduced by Brigo and Alfonsi in [3]. According to this scheme, the discrete values of r are obtained by means of the following recursion:

$$\bar{r}_{i+1} = \left(\frac{\sigma_r \sqrt{h} \mathrm{d}\bar{W}_{r,i+1} + \sqrt{\sigma_r^2 h \mathrm{d}\bar{W}_{r,i+1}^2 + 4(\bar{r}_i + \delta h)(1 + \tilde{\alpha} h)}}{2(1 + \tilde{\alpha} h)} \right)^2$$

with $\delta = \tilde{\alpha}\tilde{\gamma} - \sigma_r^2/2$.

11.4.1.2 Investigating Numerical Methods: Approximation of High-Dimensional Integrals

The main idea underlying the Monte Carlo algorithm for multivariate integration is to replace a continuous average with a discrete one over randomly selected points: the integral (11.15) is approximated by a weighted sum of N function evaluations

$$I(f) \approx I_N(f) = \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_i)$$

with weights given by $1/N$ and nodes $\mathbf{x}_i \in [0,1]^{kn}$ uniformly distributed pseudo-random sequences, where k denotes the number of risk sources. It is well known that the expected error in Monte Carlo method is proportional to the ratio $\sigma_f/\sqrt{N}$, where σ_f^2 is the variance of the integrand function and N is the number of computed trajectories. In this formula, the value σ_f depends on the integrand function and thus on the dimension of the integral, but the factor $1/\sqrt{N}$ does not. In particular, the $O(1/\sqrt{N})$ convergence rate holds for every dimension. This shows why MC becomes more and more attractive as the dimension of integral increases, in comparison with deterministic methods for numerical integration that are conversely characterized by a rate of convergence strongly decreasing with respect to the dimension.

On the other hand, the MC method presents two deficiencies: the rate of convergence is only proportional to $N^{-1/2}$ and special care has to be taken in generating independent random points because we actually deal with pseudo-random numbers.

Since, as already pointed out, the expected error of the MC method depends on the variance of the integrand, convergence can be speeded up by decreasing the variance. One of the simplest and most widely used variance reduction techniques is the *antithetic variates* (AV) [24]. This method attempts to reduce variance by introducing negative dependence between pair of replications; in particular, in a simulation driven by independent standard normal variables Z_i, this technique can be implemented by pairing the sequence Z_i with the sequence $-Z_i$. If the Z_i are used to simulate the increments of a Brownian path, then the $-Z_i$ simulate the increments of the reflection of the Brownian path about the origin: this suggests that it can result in a smaller variance. Following the antithetic variates method, an expected value $\mathbf{E}(Y)$ can be computed by generating a sequence of pair of observations $(Y_i, \tilde{Y}_i)$

Table 11.1 Parameters for the stochastic processes for risk

	$t = 04/01/1999$
$r(t)$	0.0261356909
$\tilde{\alpha}$	0.0488239077
$\tilde{\gamma}$	0.1204548842
σ_r	0.1056548588
d	0
σ_S	0.25

such that two pairs $(Y_i, \tilde{Y}_i), (Y_j, \tilde{Y}_j)$, $i \neq j$ are independent, identically distributed, while, within each pair, the observations have the same distribution but are not independent.

The use of the antithetic variates method can approximately double the computational complexity with respect to a classical Monte Carlo simulation, since, for each trajectory, two realizations of the Brownian path have to be simulated. Therefore, its application is effective if we obtain an estimator with a variance smaller than the one corresponding to a classical Monte Carlo simulation performed with a double number of trajectories, that is, if

$$\mathrm{Var}\left(\frac{1}{N}\sum_{i=1}^{N}\frac{Y_i + \tilde{Y}_i}{2}\right) < \mathrm{Var}\left(\frac{1}{2N}\sum_{i=1}^{2N} Y_i\right).$$

As will be shown later through some numerical experiments, nearly the same accuracy obtained by a classical MC method on N replications can be reached using the antithetic variates with less than $N/2$ pair of replications.

For brevity, in the following, we refer to the antithetic variates reduction technique combined with MC method as the *antithetic variates method*.

11.4.1.3 Numerical Experiments

In order to analyze the impact of different SDEs solvers, we here consider a single insurance contract and a simulated investment portfolio. CPI is not taken into account. The return of the segregated fund F_t is defined by a trading strategy on stocks and bonds

$$F_t = \delta S_t + (1 - \delta) B_t, \tag{11.17}$$

where S_t is a stock index, B_t is a bond index, and $\delta = 10\%$ at time 0.

We point out that, since the discussion follows the stages of our activity, numerical experiments refer to different dates as for processes calibration. In particular, this test case refers to the date January 4th 1999 for the evaluation of bond market. In Table 11.1 the parameters of the equations governing the considered risk sources, that is, the short rate and the stock index, are reported.

Bond market data have been estimated following [38]. All the experiments refer to a $T = 20$ year term policy for a 30 year old insured. The technical interest rate

Table 11.2 Values of $V(0;Y_T)$ (11.5)

N	Expected value	Monte Carlo	Antithetic variates
1,000	85.530725	85.330736	85.538849
2,500	85.530725	85.446784	85.513984
5,000	85.530725	85.490321	85.526349
10,000	85.530725	85.515832	85.529525
20,000	85.530725	85.556925	85.532012
50,000	85.530725	85.531398	85.532053
100,000	85.530725	85.535615	85.532456

N is the number of simulated trajectories; in the second column the expected value, that is, the sample mean computed via AV with $N = 20 \times 10^6$ is reported

is set to 4%, the yearly minimum guarantee is supposed to be $\delta^c = i$ and $\psi = 0.8$. The initial capital is set to $Y_0 = 100$. The values of the expectation of life have been computed by the life tables SIM81. Finally, the correlation coefficient between short rate and stock market is set to -0.1.

We recall that the main computational kernels are multidimensional integrals and SDEs. We focus on them separately.

In the discussion about techniques for the numerical computation of multidimensional integrals, we use the Euler scheme for the solution of the involved SDEs. We test and compare performances of MC and antithetic variates methods. In our simulations the routine *snorm* of the package **ranlib**, written by Brown, Lovato and Russell, available through Netlib repository, has been used to generate standard normally distributed values. In Table 11.2 we report the values of the undertaking labilities $V(0;Y_T)$ estimated via the two integration methods, for different values of the number N of simulated trajectories.

In order to estimate the error, an "almost true" values is needed: we assume as true expected value the sample mean computed via antithetic variates method with a number of replications equal to 20×10^6. We observe that with the classical MC method we obtain three significant digits for $N \geq 10^4$; applying antithetic variates method the same accuracy is reached just for $N = 10^3$. Moreover, to obtain four significant digits with antithetic variates we need $N = 2 \times 10^4$ simulations while with MC we need $N = 5 \times 10^4$ simulations. Since the application of antithetic variates technique at most doubles the computational cost, we deduce that efficiency is strongly improved in these cases. All the experiments confirmed this.

It is well known that special care has to be taken in generating pseudo-random points. To show the sensitivity of MC and antithetic variates methods to the initial seed of the pseudo-random number generator, in Table 11.3, we report the minimum and the maximum values of $V(0,Y_T)$ estimated via Monte Carlo method and antithetic variates, repeating each MC and antithetic variates run 20 times. We observe that MC exhibits a sensitivity greater than the one shown by antithetic variates to the seed and that both minimum and maximum estimations computed by antithetic variates deliver two significant digits for all considered values of N.

Table 11.3 To valuate the sensitivity of MC and AV to initial seed we repeat each run 20 times

	MC		AV	
N	$V(0;Y_T)$ min	$V(0;Y_T)$ max	$V(0;Y_T)$ min	$V(0;Y_T)$ max
1,000	84.448579	86.314370	85.098095	85.880454
2,500	84.962291	86.099855	85.387740	85.704434
10,000	85.097654	86.042048	85.431023	85.605205

In the 2nd and 4th column are reported the minimum resulting values for $V(0;Y_T)$, while in the 3rd and 5th column are reported the maximum resulting values

Table 11.4 MC and AV CPU times in seconds for different values of the number N of trajectories and the ratio between them

N	MC	AV	$\frac{AV}{MC}$
1,000	5.84	7.87	1.34
2,500	17.75	19.73	1.11
5,000	32.90	39.53	1.20
10,000	66.24	79.01	1.19
20,000	130.75	157.13	1.20
50,000	334.19	393.59	1.18
100,000	668.16	786.06	1.18

In Table 11.4 we show the CPU time spent by Monte Carlo simulation and antithetic variates methods. In order to evaluate the overhead of antithetic variates with respect to MC method we also report the ratio between the two values. We observe that the execution time of antithetic variates is never the double of execution time of Monte Carlo method, even though it requires the generation of a number of simulations that is double with respect to MC. The ratio between the two execution times is always about 1.2.

All the shown experiments reveal that the use of antithetic variates method allows to obtain the same accuracy as MC method with a number of replications that is reduced by a factor near to four. Then, a first important result is that the use of antithetic variates allows us to have a good accuracy for small number of replications, resulting in a large gain in terms of execution time.

We now turn to the numerical solution of SDEs. In order to estimate the error we refer to the *deterministic solution*, obtained neglecting the stochastic term in (11.7), since it can be shown that the expected value of $r(t)$, given its value at time 0, is

$$\mathbf{E}(r(t)) = \tilde{\gamma} - (\tilde{\gamma} - r(0))\mathrm{e}^{-\tilde{\alpha}t} = r_{\det}(t),$$

which is just the solution of the SDE obtained by (11.7) when the deterministic term is the only considered.

To confirm our previous statements on the better performance of the antithetic variates method over MC, we represent, in Figs. 11.1 and 11.2, the values of the absolute errors of the interest rates computed at each time with the four different SDE methods and with both MC and antithetics variates methods, for $N = 5 \times 10^3$ and for $N = 5 \times 10^4$, respectively. A monthly discretization step size is used. We

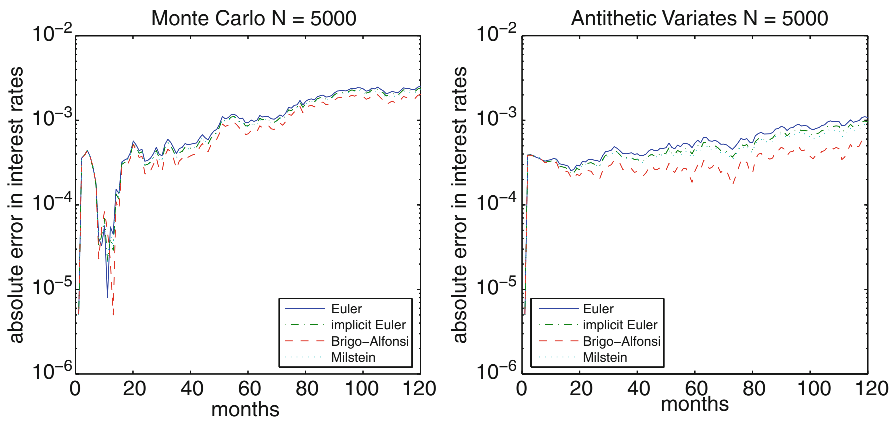


Fig. 11.1 Absolute errors in the estimation of interest rates computed with MC and antithetic variates vs. time. $N = 5 \times 10^3$ trajectories have been simulated; the discretization step size is monthly. At each time t the average over trajectories is represented

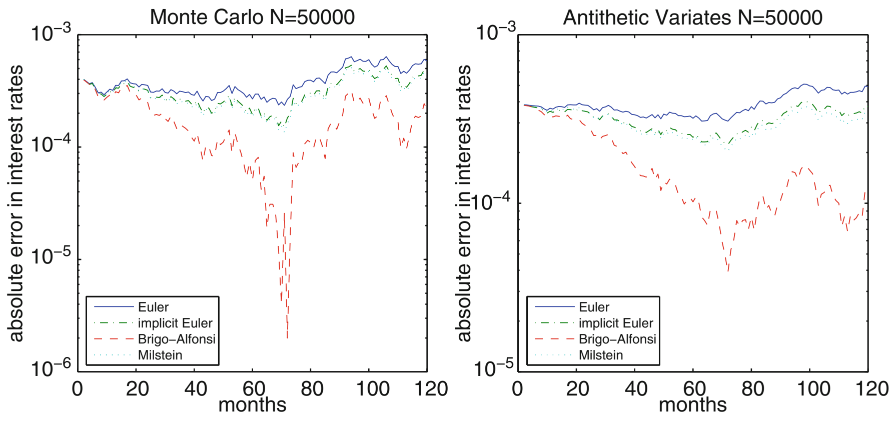


Fig. 11.2 Absolute errors in the estimation of interest rates computed with MC and antithetic variates vs. time. $N = 5 \times 10^4$ trajectories have been simulated; the discretization step size is monthly. At each time t the average over trajectories is represented

observe that, as we expected, antithetic variates outperforms from the accuracy point of view MC method in the estimation of the spot interest rates too.

In Fig. 11.3, we plot the values obtained with $N = 5 \times 10^3$ replications of MC and with $N = 10^3$ replications of antithetic variates method; we observe that errors estimated via antithetic variates with $N = 10^3$ are almost always lower than those estimated via MC with $N = 5 \times 10^3$. Analyzing now the behavior of the four different methods we note that the absolute error lies for all the methods in the interval $]10^{-6}, 10^{-3}[$. In particular, for values of N equal to 1,000 and 5,000 the accuracy given by the four methods is almost comparable. Increasing the number of

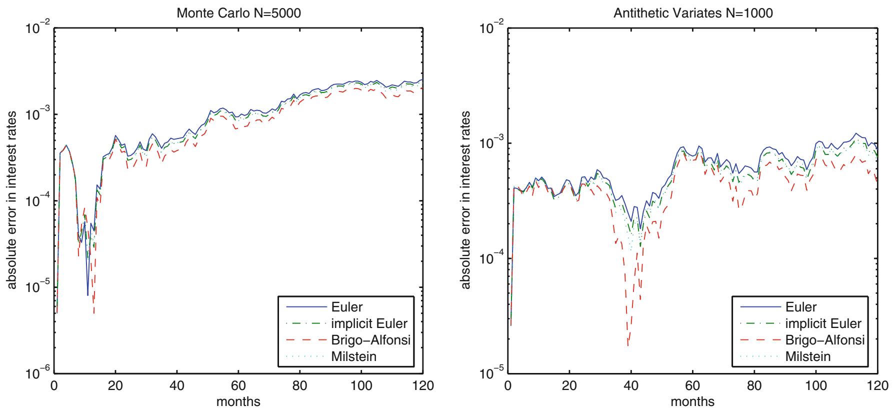


Fig. 11.3 Absolute errors in the estimation of interest rates computed with MC for $N = 5{,}000$ and with antithetic variates for $N = 1{,}000$ vs. time; the discretization step size is monthly. At each time t the average over trajectories is represented

Table 11.5 Execution times in seconds for AV method with the four different scheme for the SDEs

	Antithetic variates execution times (seconds)						
N	1,000	2,500	5,000	10,000	20,000	50,000	100,000
Euler	7.87	19.73	39.53	79.01	157.13	393.59	786.06
Implicit Euler	8.30	20.80	41.63	83.69	166.34	414.45	829.81
Brigo–Alfonsi	8.65	21.67	43.30	86.69	173.52	433.25	868.01
Milstein	9.98	25.04	49.89	99.77	199.39	498.56	1,000.37

simulations (Fig. 11.2) the Euler method exhibits the worst behavior; Implicit Euler is comparable with Milstein scheme, but the computational complexity of the latter is higher, while the Brigo–Alfonsi method reaches the highest level of accuracy.

In Table 11.5 we report the execution times for antithetic variates method using the four different schemes for the SDEs. The results show that the Milstein scheme is more time-consuming than the other ones; the Brigo–Alfonsi method is slightly more expensive than the Implicit Euler one.

Finally, in Fig. 11.4 are reported, for MC and antithetic variates methods, the values of the root-mean-square (RMS) absolute error defined by

$$\text{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{r}_i - r_{\text{det}}(t_i))^2}.$$

Figure 11.4 shows that the RMS errors of all the considered methods significantly reduce using the antithetic variates method, and Implicit Euler and Milstein have comparable behavior, while the Brigo–Alfonsi method outperforms all the other ones, especially for high values of N.

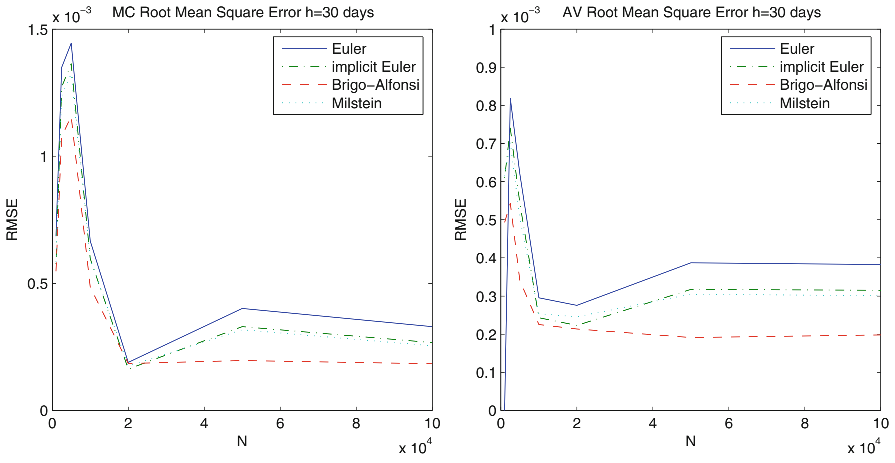


Fig. 11.4 On the *left*: RMS errors in the SDEs solution with MC method vs. N; on the *right*: RMS errors in the SDEs solution with antithetic variates vs. N. The discretization step size is monthly

11.4.2 Improving Accuracy and Efficiency: The Forward Risk-Neutral Measure

From this section on, we deal with the numerical simulation of a real ALM portfolio; we specifically refer to Dynamic Investment Strategy with Accounting Rules (DISAR), an ALM system for the valuation and risk management of portfolios of profit-sharing policies [8]. The methodological ALM framework in which DISAR has been designed is detailed in [9].

The simulation of an asset–liability portfolio results in a large-scale computational problem. Let 0 and T be, respectively, the beginning and the end of the simulation period, expressed in years; we consider, as before, a time grid with $n+1$ equally spaced points in $[0,T]$. Let $h>0$ be the fixed time step, generally equal to 1 month. According to the ALM strategy, then, every month the investment strategy is reviewed in dependence from the current value of assets and liabilities. Therefore, all the involved quantities, expressed by complex functions of random variables, have to be evaluated at each month up to year T. Moreover, at least at the end of each year, the balance sheet and the statutory reserve have to be computed. In Fig. 11.5 an outline of a procedure for ALM of a portfolio is shown.

Typically, these simulations require hours to be performed [8]. We consider a change of *numéraire* in the stochastic processes for risk sources, since the flexibility of this approach can be particularly valuable in a model with stochastic interest rates. In particular, we consider the numéraire which defines the *forward risk-neutral measure*. This choice is motivated by the observation that pricing under the forward measure can provide considerable gains in accuracy, since it allows to discount at a deterministic price deflator, even though the short rate is stochastic: indeed

```
begin ALM portfolio
% $N$ is the number of trajectories in MC scheme
% $T$ is the end of the simulation period
for i = 1 to N do
    simulate risk factors ⇔ integrate (11.7), (11.11), (11.12) in [0,$T$]
    for each month:
        evaluate assets and liquidity
        apply the investment strategy
        revaluate the insured capitals according to (11.1)
    for each year:
        evaluate the balance sheet
        evaluate the statutory reserve
endfor
compute the averages over the trajectories
end
```

Fig. 11.5 Sketch of a procedure for the numerical simulation of asset–liability portfolio of PS policies

the forward measure is considered the right probability measure when evaluating a future random cash flow in a stochastic interest rate environment [23].

The forward risk-neutral measure for maturity T (the expression has been proposed by Jamshidian in [28]) is the probability measure associated with taking as numéraire a zcb maturing at T, with unitary face value [4,23,24,28]. In this case, it can be shown that the pricing formula of a security becomes

$$V(t) = B(t,T)\mathbf{E}^F\left[V(T)\right],$$

where $B(t,T)$ is the value in t of the numéraire bond, having expression (11.8) when the short rate evolves according to the CIR model. $\mathbf{E}^F$ denotes expectation under the forward measure. As a consequence, $V(0,T_Y)$ under the forward measure can be computed as

$$V(0,Y_T) = Y_0 B(0,T)\mathbf{E}^F\left[\Phi_T\right]{}_T p_x. \tag{11.18}$$

The bond price (11.8) dynamics is given, under the risk-neutral measure, by [24]

$$\frac{\mathrm{d}B(t,T)}{B(t,T)} = r(t)\mathrm{d}t - \beta_r(t,T)\sigma_r\sqrt{r(t)}\mathrm{d}\tilde{W}_r(t)$$

with β_r defined in (11.10). Applying Girsanov's theorem, it can be shown that the process W_r^F defined by

$$\mathrm{d}W_r^F = \mathrm{d}\tilde{W}_r + \sigma_r\sqrt{r(t)}\beta_r(t,T)\mathrm{d}t$$

is a standard Brownian motion under the forward measure. It can be proved that, in general, a change of numéraire affects the drift of the processes only [4,24].

The components of the vector

$$\mathbf{W}^F = (W_r^F, W_p^F, W_S^F),$$

defined by

$$\begin{aligned} \mathrm{d}W_r^F(t) &= \mathrm{d}\tilde{W}_r(t) + \sigma_r\sqrt{r(t)}\beta_r(t,T)\mathrm{d}t, \\ \mathrm{d}W_p^F(t) &= \mathrm{d}\tilde{W}_p(t), \\ \mathrm{d}W_S^F(t) &= \mathrm{d}\tilde{W}_S(t) \end{aligned} \tag{11.19}$$

are independent Brownian motions under the forward measure [4]. From (11.19) it follows

$$\begin{aligned} \mathrm{d}\tilde{W}_r(t) &= \mathrm{d}W_r^F(t) - \sigma_r\sqrt{r(t)}\beta_r(t,T)\mathrm{d}t, \\ \mathrm{d}\tilde{W}_p(t) &= \mathrm{d}W_p^F(t), \\ \mathrm{d}\tilde{W}_S(t) &= \mathrm{d}W_S^F(t). \end{aligned} \tag{11.20}$$

Combining (11.13) and (11.20), we obtain

$$\mathrm{d}\bar{\mathbf{W}} = \mathbf{L}\cdot\mathrm{d}\tilde{\mathbf{W}} = \begin{pmatrix} \mathrm{d}W_r^F(t) - \sigma_r\sqrt{r(t)}\beta_r(t,T)\mathrm{d}t \\ \mathrm{d}W_p^F(t) \\ l_{13}(\mathrm{d}W_r^F(t) - \sigma_r\sqrt{r(t)}\beta_r(t,T)\mathrm{d}t) + l_{33}\mathrm{d}W_S^F(t) \end{pmatrix}.$$

The dynamics of the correlated state variables becomes, thus, under the forward risk-neutral measure

$$\begin{aligned} \mathrm{d}r(t) &= \tilde{\alpha}[\tilde{\gamma} - r(t)]\,\mathrm{d}t + \sigma_r\sqrt{r(t)}\,[\mathrm{d}W_r^F(t) - \sigma_r\sqrt{r(t)}\beta_r(t,T)\mathrm{d}t], \\ \frac{\mathrm{d}p(t)}{p(t)} &= \tilde{y}_t\mathrm{d}t + \sigma_p\mathrm{d}W_p^F(t), \\ \frac{\mathrm{d}S(t)}{S(t)} &= (r(t)-d)\mathrm{d}t + \sigma_S[l_{13}(\mathrm{d}W_r^F(t) - \sigma_r\sqrt{r(t)}\beta_r(t,T)\mathrm{d}t) + l_{33}\mathrm{d}W_S^F(t)], \end{aligned}$$

from which it follows

$$\begin{aligned} \mathrm{d}r(t) &= [\tilde{\alpha}\tilde{\gamma} - (\tilde{\alpha} + \sigma_r^2\beta_r(t,T))r(t)]\,\mathrm{d}t + \sigma_r\sqrt{r(t)}\,\mathrm{d}W_r^F(t), \\ \frac{\mathrm{d}p(t)}{p(t)} &= \tilde{y}_t\mathrm{d}t + \sigma_p\mathrm{d}W_p^F(t), \\ \frac{\mathrm{d}S(t)}{S(t)} &= [r(t) - d - l_{13}\sigma_S\sigma_r\sqrt{r(t)}\beta_r(t,T)]\mathrm{d}t + \sigma_S[l_{13}\mathrm{d}W_r^F(t) + l_{33}\mathrm{d}W_S^F(t)]. \end{aligned}$$

Let

$$\bar{\mathbf{W}}^F = (\bar{W}_r^F, \bar{W}_p^F, \bar{W}_S^F)$$

be the correlated Brownian motion under the forward measure; the following relation holds:

$$\mathrm{d}\bar{\mathbf{W}}^F = \mathbf{L}\cdot\mathrm{d}\mathbf{W}^F = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_{13} & 0 & l_{33} \end{pmatrix} \cdot \begin{pmatrix} \mathrm{d}W_r^F(t) \\ \mathrm{d}W_p^F(t) \\ \mathrm{d}W_S^F(t) \end{pmatrix} = \begin{pmatrix} \mathrm{d}W_r^F(t) \\ \mathrm{d}W_p^F(t) \\ l_{13}\mathrm{d}W_r^F(t) + l_{33}\mathrm{d}W_S^F(t) \end{pmatrix}.$$

The forward risk-neutral dynamics of the correlated state variables is therefore

$$\begin{aligned} \mathrm{d}r(t) &= [\tilde{\alpha}\tilde{\gamma} - (\tilde{\alpha} + \sigma_r^2\beta_r(t,T))r(t)]\,\mathrm{d}t + \sigma_r\sqrt{r(t)}\,\mathrm{d}\bar{W}_r^F(t), \\ \frac{\mathrm{d}p(t)}{p(t)} &= \tilde{y}_t\mathrm{d}t + \sigma_p\mathrm{d}\bar{W}_p^F(t), \\ \frac{\mathrm{d}S(t)}{S(t)} &= [r(t) - d - l_{13}\sigma_S\sigma_r\sqrt{r(t)}\beta_r(t,T)]\mathrm{d}t + \sigma_S\mathrm{d}\bar{W}_S^F(t). \end{aligned}$$

Note that the dynamics of the state variables under the forward measure depends on the bond maturity T. In ALM models, where the valuation of $V(0,Y_T)$, by means of (11.18), has to be performed at least for all $T \in \mathbf{t}$, that is, at least at all the payment dates, this dependence implies that the simulation process of the dynamics of state variables changes with respect to the specific date of payment. This affects the computational complexity of the numerical evaluation process. Nevertheless, pricing under the forward measure can provide a very strong reduction of variance in the estimates of expected values, thus fewer trajectories have to be simulated with a consequent overall simulation time reduction, in spite of the mentioned computational complexity increase. In the following, we show this by means of numerical results.

11.4.2.1 Numerical Experiments

The insurance portfolio we consider contains about 78,000 policies aggregated in 5,600 fluxes. The time horizon of simulation is 40 years. The segregated fund includes about 100 assets, both bonds and equities. The return of the segregated fund F_t is defined by the trading strategy introduced in (11.17) on stocks and bonds, with $\delta = 5.08\%$ at time t_0.

We solve the SDEs for the risk sources by means of the Euler method with a monthly discretization step; as a consequence the dimension of the involved integrals is $3\cdot 480$.

The CIR model for short rate is calibrated on market data at December 30th 2005. In Table 11.6 the parameters for the stochastic processes for risks (11.7), (11.11), and (11.12) are reported. The values of the expectation of life are computed by means of the life tables SIM92. Finally, the correlation factor between $\mathrm{d}\tilde{W}_r$ and $\mathrm{d}\tilde{W}_S$ is set at 0.07.

Table 11.6 Parameters for the stochastic processes for risk

	$t = 12/30/2005$
$r(t)$	0.0261356909
$\tilde{\alpha}$	0.3240399040
$\tilde{\gamma}$	0.0393541850
σ_r	0.0534477680
d	0.027
σ_S	0.1
y_0	0.0236
y_∞	0.0211
σ_p	0.0089
α_y	2.897

Table 11.7 Column 1: number of MC trajectories; column 2: probability measure; column 3: 95% confidence interval for outstanding company liabilities; column 4: half-width of the interval

95% confidence intervals				
N	Measure	$\hat{V} - z_{0.05/2}\frac{s}{\sqrt{N}}$	$\hat{V} + z_{0.05/2}\frac{s}{\sqrt{N}}$	$z_{0.05/2}\frac{s}{\sqrt{N}}$
1,000	R-N	[2,305,030,474	2,311,939,670]	3,454,598
	FW	[2,308,799,862	2,309,475,980]	338,059
2,000	R-N	[2,305,212,676	2,310,184,153]	2,485,739
	FW	[2,308,871,978	2,309,341,618]	234,820
4,000	R-N	[2,306,464,218	2,310,001,768]	1,768,775
	FW	[2,308,776,199	2,309,104,117]	163,959
5,000	R-N	[2,306,272,289	2,309,443,908]	1,585,810
	FW	[2,308,784,101	2,309,076,514]	146,206

The experiments shown here and in the following sections have been carried out on an IBM Bladecenter installed at *Università di Napoli "Parthenope."* It consists of 6 Blade LS 21, each one of which is equipped with 2 AMD Opteron 2210 and with 4 GB of RAM.

In Table 11.7 we compare the 95% confidence intervals obtained in the estimation of $V(0, Y_T)$, when stochastic processes for risks are modeled under the risk-neutral and the forward measure, respectively, for different values of Monte Carlo simulated trajectories. We denote by $\hat{V}$ the sample mean, and, as usual, by $z_{0.05/2}$ the 95% quantile of the standard normal distribution, by s the sample standard deviation, so that $s/\sqrt{N}$ is the standard error. In all the cases, we observe that the confidence intervals obtained via the forward measure are contained into the corresponding ones estimated under the risk-neutral measure; moreover, the half-width of the confidence intervals estimated in the risk-neutral setting is about ten times the half-width observed working under the forward measure, thus we gain one order of magnitude in terms of accuracy.

In Fig. 11.6 we show the relative standard error

$$\mathrm{RSE} = \frac{s}{\hat{V}\sqrt{N}}.$$

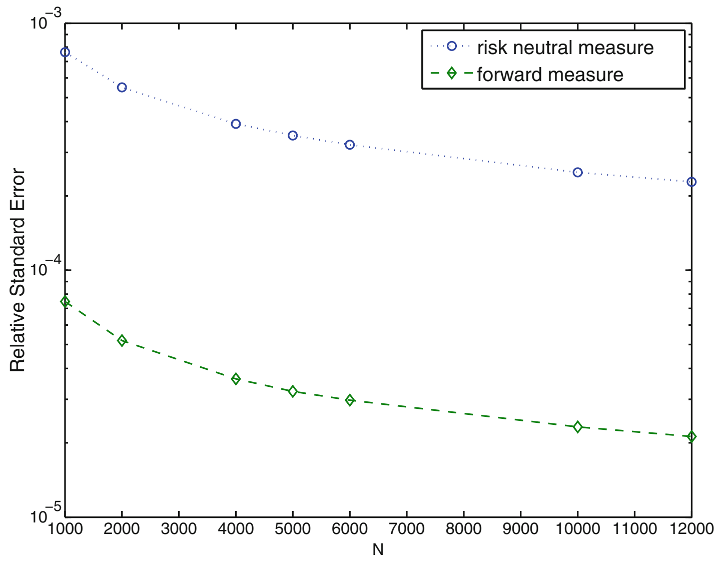


Fig. 11.6 RSE vs. number of MC trajectories

The two lines representing the RSE exhibit almost the same slope, that is, an almost constant reduction factor in RSE is observed when simulating under the forward measure, coherently to values reported in Table 11.7. On the other hand, we recall that, as already pointed out in Sect. 11.4.2, the dynamics of the state variables under the forward measure depends on the bond maturity T. Therefore, for each evaluation date, a different simulation has to be carried out for all the risk sources. This obviously results in a time overhead, as it can be seen in Fig. 11.7, where the execution time, expressed in hours, for different values of simulated trajectories is reported.

In order to go deep inside into the matter, in Table 11.8 we report the values of RSE obtained in the risk-neutral setting, in the forward setting, and the ratio between them, for different numbers of MC simulated trajectories. We note that, as already observed, the ratio between the RSE values is always about ten; the RSE obtained when working under the forward measure with $N = 1{,}000$ MC trajectories is smaller than the one concerning the MC simulation under risk-neutral measure with $N = 12{,}000$, thus confirming that modeling risks under the forward measure results in a considerable improvement in terms of accuracy. In Table 11.8 we also report the execution time, expressed in hours, required in the two cases and the ratio between these values, which gives the time overhead related to the forward measure approach. We note that the simulation under the forward measure almost doubles the execution time with respect to the one performed under the risk-neutral measure, but we observe that the simulation corresponding to $N = 1{,}000$ trajectories under the forward measure provides, in about 17 min, more accurate estimates than the one corresponding to $N = 12{,}000$ trajectories under the risk-neutral measure

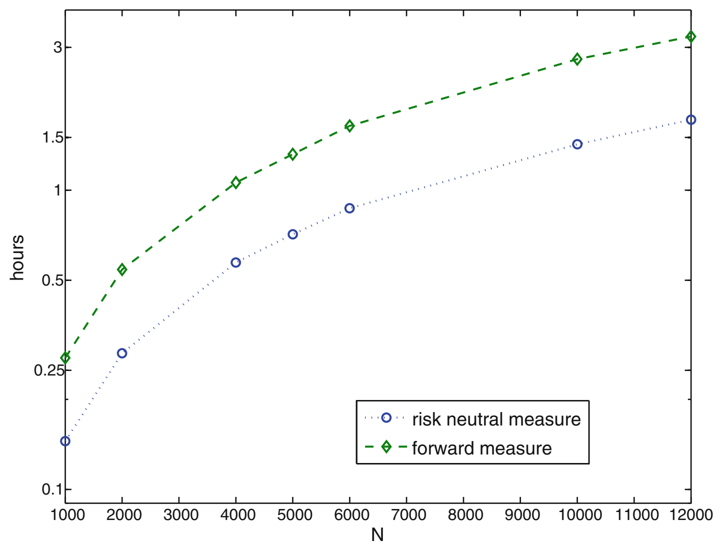


Fig. 11.7 Execution time in hours vs. number of MC trajectories

Table 11.8 Column 1: number of MC trajectories; column 2: RSE under the risk-neutral measure; column 3: RSE under the forward measure; column 4: RSE under the risk-neutral measure over RSE under the forward measure. Column 5: execution time in hours under the risk-neutral measure; column 6: execution time in hours under the forward measure; column 7: overhead of the forward measure approach

	RSE			Execution time		
N	RNM	FM	Ratio	RNM	FM	Ratio
1,000	7.635E-04	7.469E-05	10.22	0.15	0.28	1.90
2,000	5.496E-04	5.188E-05	10.59	0.29	0.54	1.90
4,000	3.910E-04	3.623E-05	10.79	0.57	1.06	1.85
5,000	3.506E-04	3.231E-05	10.85	0.71	1.32	1.85
6,000	3.219E-04	2.976E-05	10.82	0.87	1.64	1.90
10,000	2.486E-04	2.310E-05	10.74	1.42	2.74	1.92
12,000	2.278E-04	2.121E-05	10.74	1.72	3.26	1.89

as well, which requires about 1 h and 45 min. This means that the forward risk-neutral approach is only apparently more time-consuming, since it actually requires a significantly smaller number of Monte Carlo trajectories to provide the same accuracy as the risk-neutral approach.

11.4.3 *Improving Accuracy and Efficiency: Introducing Parallelism*

In this section we discuss the development of a parallel algorithm for the simulation of PS policies.

```
Procedure PMC(in: $N$; $P$; $f$; out: $I(f)$)
    %initialize parallel processes
    %generate distributed pseudo-random sequences
    %$\mathbf{x}_i^j$, $j = 0, \ldots, P-1$, $i = 1, \ldots, N/P$ is the local stream distributed to proc. $j$
    % each processor $j$ computes the average over local trajectories
    for $i = 1, N/P$ do
        $I_{N/P}^j(f) = \frac{1}{N/P} \sum_{i=1}^{N/P} f(\mathbf{x}_i^j)$
    endfor
    %combine local averages
    $I_N(f) = \frac{1}{P} \sum_{j=0}^{P-1} I_{N/P}^j(f)$
End Procedure
```

Fig. 11.8 Parallel Monte Carlo algorithm for the valuation of participating life insurance policies

We introduce parallelism by means of a parallel Monte Carlo algorithm. The natural strategy for parallelizing Monte Carlo method is to distribute trajectories; in distributed environments this means that trajectories are distributed among processors.

If N trajectories are generated in a parallel simulation involving P processors, each processor computes the average over N/P randomly selected points, then the partial results are combined to obtain the overall sample mean. Therefore, Monte Carlo is generally considered "naturally parallel." In Fig. 11.8 a sketch of a typical parallel MC algorithm is represented. However, the core of MC is the generation of pseudo-random sequences capable to mimic random samples drawn from uniform distribution. Effective pseudo-random generators (PRG) must provide long-period sequences of uncorrelated values. The effectiveness of a parallel MC algorithm is strongly related to the relying parallel pseudo-random generator (PPRG) as well: even though MC is generally intended inherently parallel, parallelism requires special care in the use of PRG. Generation of pseudo-random sequences in a parallel setting must deal with both inter-processor and intra-processor correlations. Indeed, values belonging to a pseudo-random subsequence distributed to a processor must be uncorrelated; moreover, correlation among numbers distributed to different processors is undesirable too. Moreover, efficiency is a critical issue as well, thus, inter-processor communication should be as minimum as possible. Parallelization schemes based on *cycle parametrization* are mostly employed. These methods rely on the capability of certain generators to produce different full-period streams, that is, nonoverlapping sequences, given different, carefully chosen, seeds. In this way, processors concurrently generate uncorrelated streams, thus providing scalable procedures [35].

We developed a parallel MC algorithm, employing a parallel ALFG based on a cycle parametrization technique [34]. In Fig. 11.9 the outline of the parallel MC algorithm is shown. Communication among processors is limited to the initialization phase, where basic common information are to be exchanged, and the final phase, when partial results are to be combined to compute the global average which gives the MC method result.

```
Procedure PMC_CP(in: $N$; $P$; $f$; $id$; out: $I(f)$)
    %initialize parallel processors
    %processor $id$ generates its local stream
    generate $\mathbf{x}_1^{id}, \ldots, \mathbf{x}_{N/p}^{id}$
    %processor $id$ computes the average over local trajectories
    for $i = 1, N/P$ do
        $I_{N/P}^{id}(f) = \frac{1}{N/P} \sum_{i=1}^{N/P} f(\mathbf{x}_i^{id})$
    endfor
    %combine local averages
    $I_N(f) = \frac{1}{P} \sum_{j=0}^{P-1} I_{N/P}^{j}(f)$
End Procedure
```

Fig. 11.9 Parallel Monte Carlo algorithm based on cycle parametrization technique for the generation of pseudo-random streams

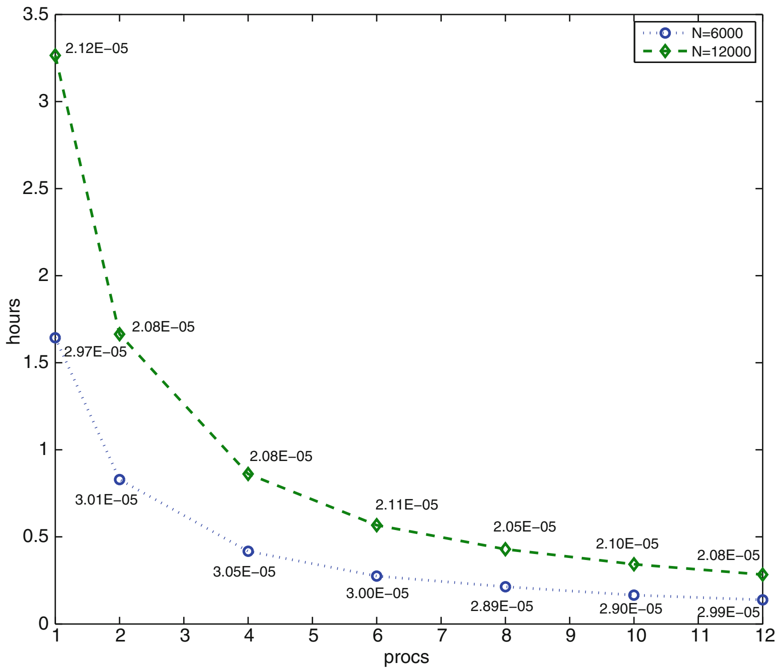


Fig. 11.10 Execution time in hours vs. number of processors involved in the simulation. The global number of simulated trajectories is fixed. For each simulation, the value of the RSE is also reported

Since the change of numéraire does not affect parallel performances, we confine our discussion to the forward measure, which, as already pointed out, gives the more accurate results. In Fig. 11.10 the execution time in hours versus the number of processors involved in the computation is represented. Here the global number

Table 11.9 Execution time in minutes; the global number of trajectories is fixed

nprocs	$N = 6{,}000$	$N = 12{,}000$
1	92	196
2	50	100
4	25	52
6	17	34
8	13	26
10	10	21
12	8	17

of trajectories is fixed; for this reason, the number of MC simulated trajectories has been chosen so to be divisible by the number of processors. Moreover, we report for each simulation time the corresponding RSE value. The execution time values, expressed in minutes, are also reported in Table 11.9 for the sake of readability. We observe that the RSE keeps the same order of magnitude as the number of processors increases, thus confirming the scalability of the chosen parallel pseudo-random number generator. In this case, if we fix a target accuracy for estimates, then parallelism allows to realize the target accuracy in a strongly reduced time: this is clearly meaningful to insurance undertaking.

In Fig. 11.11 the RSE versus the number of processors involved in the computation is shown. The results refer to simulations in which we fix the local number of simulated trajectories: thus, for instance, the point corresponding to four processors in the line referring to $N = 1{,}000$ trajectories is the value of the RSE obtained with a global number of 4,000 trajectories. Moreover, we report for each estimated RSE value the related execution time in hours. Looking at the two lines, we observe that the execution times at most vary on the second decimal digit with respect to processors, thus confirming the scalability of the parallel MC algorithm, and, obviously, the RSE is reduced. Therefore, if we fix a target time for responses, then, parallelism allows to improve the estimate reliability within the target time.

We finally show, in Fig. 11.12, the speedup for the same simulations. The graphic reveals the good scalability properties of the algorithm. The same behavior was observed in all our experiments. The speedup is almost linear: this is motivated by the lowest communication cost of the parallel algorithm, indeed, communication is only required during the initialization phase of the pseudo-random number generator and during the reduction phase, when the local averages are combined to compute the sample mean provided by the Monte Carlo method.

11.5 Default-Risk Modeling

In this section we deal with default-risk modeling; the reason for this is the increasing interest on this topic; the Solvency II Directive establishes that “insurance and reinsurance undertakings may take full account of the effect of risk-mitigation techniques in their internal model, as long as credit risk ... ” (art. 121). “The credit risk”

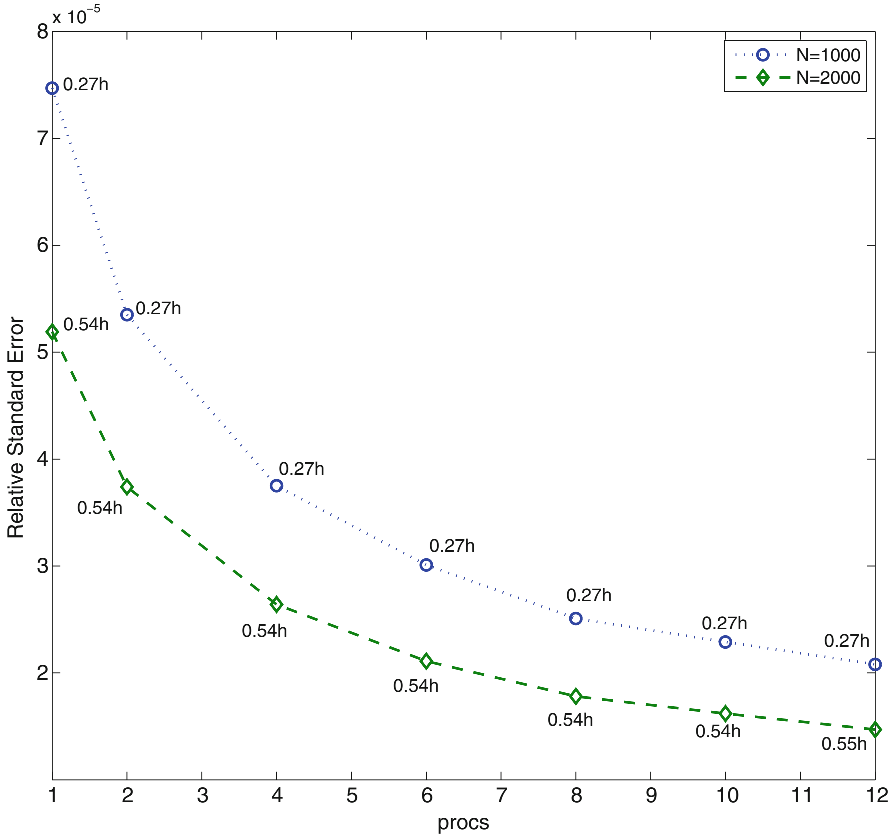


Fig. 11.11 RSE vs. number of processors involved in the simulation. The local number of simulated trajectories is fixed. For each simulation, the value of the execution time in hours is also reported

is defined as "the risk of loss or of adverse change in the financial situation, resulting from fluctuations in the credit standing of issuers of securities, counterparties and any debtors to which insurance and reinsurance undertakings are exposed, in the form of counterparty default risk, or spread risk, or market risk concentrations" (art. 13(32)). From the definition above, it is clear that default risk is a relevant component of credit risk. Specifically, in profit-sharing life insurance policies a part of the counterparty default risk falls on the insurance undertaking, because of the minimum guarantees, and a part is transferred to the policyholders, by the retrocession mechanism.

Default-risk modeling mainly follows two approaches, one leading to the so-called *structural models*, another, more recently become popular, leading to *reduced-form models*. The former has its foundations in option pricing techniques, following Black and Scholes and Merton. Structural models are based on modeling the stochastic evolution of the issuer's balance sheet, with default occurring when

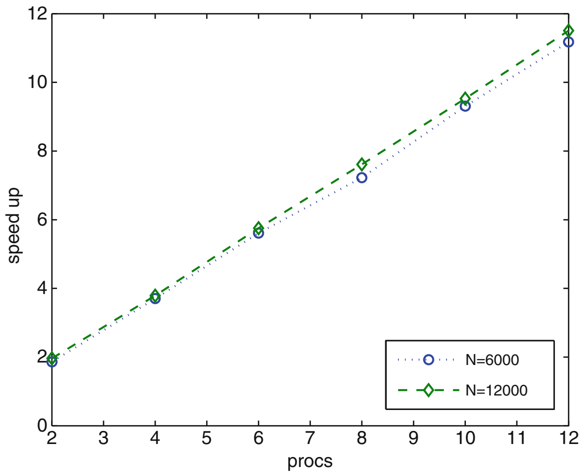


Fig. 11.12 Speedup vs. number of processors

the issuer is unable to meet its obligations. These models are generally difficult to implement for several reasons, for instance, issuer's assets and liabilities could be not traded, thus their market value could be not available [1]. Moreover, structural models do not generally allow one to capture credit spreads within a short time horizon if a continuous sample path is assumed for the asset process [21]. This can be overcome by assuming incomplete information about the asset value of the firm: in this case, the curves representing credit spreads are quite similar to the ones obtained in the case of *reduced-form models*, which have become popular in the last decade. Reduced-form models have been proven to be fully consistent with option-based approach if uncertainty on information is assumed [32]. According to them, default is treated as an unexpected event with the likelihood governed by a *default-intensity process*: the default-intensity measures the conditional likelihood that the issuer will default over the next (small) interval of time, given that it has not yet defaulted and given all other available information. Reduced-form models have attracted great interest also because this approach involves default-free term structure modeling, thus one can benefit from all existing literature [29]. In this framework moreover some results which turn out to be very useful in practice have been proven: for instance, corporate bond pricing is greatly simplified when default probability is modeled as an intensity process [20]. For all these reasons, in the following we focus on this second approach: we discuss the use of stochastic models to measure default risk in the development of internal models, according to the rules of the European Directive.

We start by introducing the modeling context we refer to. Let τ be the time of default and $\bar{B}(0,T)$ the price at time 0 of a defaultable unitary zcb with maturity T. Under the zero-recovery hypothesis, the arbitrage-free price is given by the expected value of the risk-free discount-factor conditional to survival up to time T [21]

$$\bar{B}(0,T) = \mathbf{E}^{Q}\left[\mathrm{e}^{-\int_0^T r(u)\mathrm{d}u}\mathbf{1}_{\{\tau>T\}}\right].$$

$\mathbf{1}_{\{\tau>T\}}$ is the indicator function of the event $\tau > T$. τ is modeled as the first arrival time of a Poisson process with random arrival rate λ, that is, the default event is conditional on the information given by the path of the random intensity

$$\{\lambda(u) : u \geq 0\}.$$

Therefore, a *doubly stochastic process* is considered, in that we have two layers of uncertainty, both the time and the intensity of default [21]. In this framework, the survival probability in $[0,T]$ has the following expression:

$$P(0,T) = \mathbf{E}^Q\left[\mathrm{e}^{-\int_0^T \lambda(u)\mathrm{d}u}\right].$$

A very useful result, proved by Lando [32], provides an expression for the price at time 0 of a defaultable unitary zcb in terms of the short rate and the default intensity

$$\bar{B}(0,T) = \mathbf{E}^Q\left[\mathrm{e}^{-\int_0^T r(u)+\lambda(u)\,\mathrm{d}u}\right]. \tag{11.21}$$

Moreover, r and λ are often assumed to be stochastically independent; this simplifying assumption actually appears to be a reasonable first approximation in the historical probabilities for investment-grade debt [29]. The expected value in (11.21) can be factorized in the following way [32]:

$$\bar{B}(0,T) = B(0,T)P(0,T), \tag{11.22}$$

where $B(0,T)$ is the price in 0 of the unitary default-free zcb with maturity T. Relation (11.22) states that the price of the defaultable zcb is the product of the price of the corresponding non-defaultable zcb times the "markdown" factor, given by the survival probability in the time interval delimited by the pricing time and the zcb maturity.

If the default intensity λ is supposed to follow the classical Cox, Ingersoll and Ross process, that is, λ evolves as square root diffusions, the stochastic equation governing the process is

$$\mathrm{d}\lambda(t) = \tilde{k}[\tilde{\theta} - \lambda(t)]\,\mathrm{d}t + \sigma_\lambda\sqrt{\lambda(t)}\,\mathrm{d}Z_\lambda(t). \tag{11.23}$$

So, in the described modeling framework, both r and λ are modeled as basic affine processes; under independence hypothesis, a closed form for survival probability in $[0,T]$ can be obtained, which depends on the parameters of the intensity process in (11.23) and on the value of λ at $t=0$

$$P(0,T) = A_\lambda(0,T)\mathrm{e}^{-\lambda(0)\beta_\lambda(0,T)}, \tag{11.24}$$

where

$$A_\lambda(t,T) = \left[\frac{d_\lambda e^{\phi_\lambda(T-t)}}{\phi_\lambda(e^{d_\lambda(T-t)}-1)+d_\lambda}\right]^{\nu_\lambda}, \tag{11.25}$$

$$\beta_\lambda(t,T) = \frac{e^{d_\lambda(T-t)}-1}{\phi_\lambda(e^{d_\lambda(T-t)}-1)+d_\lambda} \tag{11.26}$$

with

$$d_\lambda = \sqrt{\tilde{k}^2+2\sigma^2}, \quad \nu_\lambda = \frac{2\tilde{k}\tilde{\theta}}{\sigma^2}, \qquad \phi_\lambda = \frac{\tilde{k}+d_\lambda}{2}. \tag{11.27}$$

11.5.1 The Data and Calibration Method

Model calibration is well known to be a fundamental task in risk modeling. In this section we present a procedure for the calibration of (11.23); the problem is formulated as a nonlinear least-squares one.

Special care has to be addressed to data: in the context of default-risk modeling, an adequate set of bonds to calibrate the spread curve to has to be chosen. We start by discussing a data selection procedure.

The idea underlying data selection is to model the evolution of a collection of term structures of defaultable bond credit spread with different credit ratings and sectors.[2]

Rating are used as a way of aggregating corporate bond prices [32]. To obtain a benchmark term structure for corporate bonds it is necessary to pool different bonds and rating is a natural place to start; nevertheless ratings are not sufficient to price a bond; then, as it is usually done, we pool bonds also for different economic sectors.

We consider, in particular investment-grade bonds, the only ones that can be included in the investment portfolio of profit-sharing policies. Under technical conditions discussed before, the price, at time 0, of a zero-recovery zcb in the jth credit class of corporate bonds with maturity T is given by

$$\bar{B}_j(0,T) = E^Q\left[e^{-\int_0^T r(u)+\lambda_j(u)\,du}\right]. \tag{11.28}$$

A credit class can be interpreted as an indicator of credit quality, such as a rating category assigned to bonds by a rating agency such as Moody's or Standard & Poor's and the economic sector of bonds. Hence, $\lambda_j(u)$ is the intensity of a Poisson

[2]We do not deal with a ratings-based model, in the form of [21, 22, 31]. We do not consider the potential of upgrades or downgrades of the underlying bonds, which would result in a shift of the default intensity to that of the new rating. We use constant credit quality corporate bond indices to estimate the dynamics of the default intensities.

Table 11.10 Risk-neutral parameters for CIR model

	$t = 04/30/2010$
$r(0)$	0.00560384
$\tilde{\alpha}$	0.30368297
$\tilde{\gamma}$	0.04367279
σ_r	0.13681307

process used to model the event of default of the jth credit class. We use data on investment-grade bonds from Finance sector with Moody's rating Aa3 e Baa1. We consider current market data of a set of coupon bonds at April 30th 2010; we refer to the closing prices. For each corporate bond, the corresponding time to maturity as well as the coupon payments, the number of payments remaining, and the time to next payment plus the accrued interest as of that date have been computed using information available on Bloomberg. These data have been ordered by time to maturity.

Even within a fairly homogeneous sample, the credit spreads of some bonds can noticeably vary [40]. A problem which arises then is the removal of outliers. Outliers are bonds whose spreads deviate very much from the average spreads of the other bonds. Including them into the sample would distort the whole calibrated curve. Typically these are bonds that have not been down- or up-graded by rating agencies although the market trades them at down- or up-graded spread levels. We use a two-stage procedure to remove outliers. In the first stage a bond is removed if its yield deviates more than twice the standard deviation from the average yield in the same maturity bracket. Afterwards, the same procedure is repeated.[3]

Considering $t = 04/30/2010$ as evaluation date, we initially extracted from Bloomberg database:

- 75 Aa3 defaultable bonds of Finance sector with residual maturities ranging from 3 months up to 20 years
- 62 Baa1 defaultable bonds of Finance sector with residual maturities ranging from 3 months up to 10 years

Defaultable bond yields have two components: the risk-free interest rate and the default spread. Risk-neutral parameters for CIR model calibrated on market data are reported in Table 11.10.

We estimate the vector of default-risk parameters, $(d_\lambda, \nu_\lambda, \phi_\lambda, \lambda(0))$, by performing a nonlinear fit procedure on the two different sets of corporate bonds. The estimation is then done by means of a nonlinear least-square method that minimizes the sum of the quadratic differences between the market prices and the model prices.

In the calibration phase, the second stage of the outliers removal procedure, as described in [40], has been performed by removing those defaultable bonds whose pricing errors exceed two times the average root mean-squared relative pricing

[3]We refer to criteria applied by ECB when selecting bonds for the estimation of yield curves (www.ecb.int/stats/money/yc/html/index.en.html).

Table 11.11 Risk-neutral and Brown–Dybvig default-intensity parameters for Aa3-rated and Baa1-rated bonds at evaluation time t

$t = 04/30/2010$	Aa3-finance	Baa1-finance
$\lambda(t)$	0.00079011351155	0.00762255771740
d_λ	0.32209372929069	0.36215265757040
ϕ_λ	0.30328555876149	0.36208344131790
ν_λ	1.00000000000468	321.544667798335
sqmr	0.37561488234354	0.29840697224378

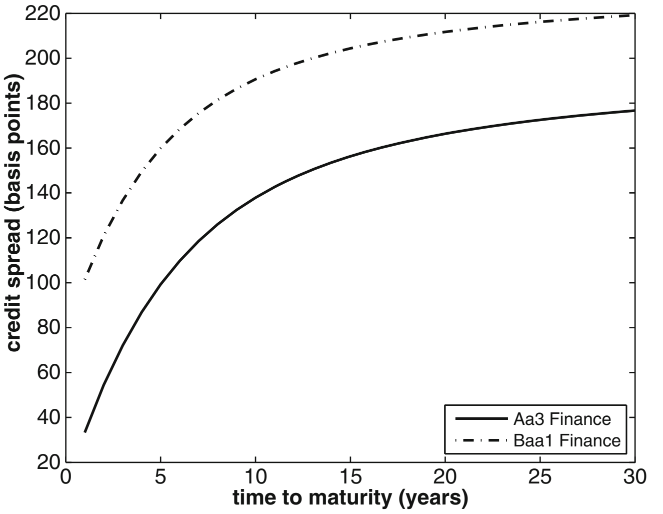


Fig. 11.13 Credit spread for Aa3-rated and Baa1-rated bonds

errors and afterwards by repeating the calibration procedure. For the implementation of the above calibration and selection procedure, we use the Matlab software environment (*trust-region reflective Newton* method).

In Table 11.11 we report the resulting estimates of the default-risk parameters in (11.24)–(11.27), using the two different sets of selected corporate bonds and the sample standard deviation of residuals.

In Fig. 11.13, for both the two credit rating classes, we plot the term structures of credit spreads (in basis points) using the calibrated default-intensity process parameters, over a range of 30 years of maturities; in Fig. 11.14 we plot the related risk-neutral default intensity $\lambda(t)$ too.

Credit spreads range from about 33 to 176 basis points for Aa3-rated bonds and from 101 to 219 basis points for Baa1 ones. In Table 11.12 we report the Credit Default Swap (CDS),[4] at the valuation date April 30th 2010, for 1, 2, 3, 4, 5, 7, 10 years to maturity, of two Aa3-rated bonds used in the calibration procedure.

[4]Source Bloomberg.

Fig. 11.14 Default intensity $\lambda(t)$ for Aa3-rated and Baa1-rated bonds

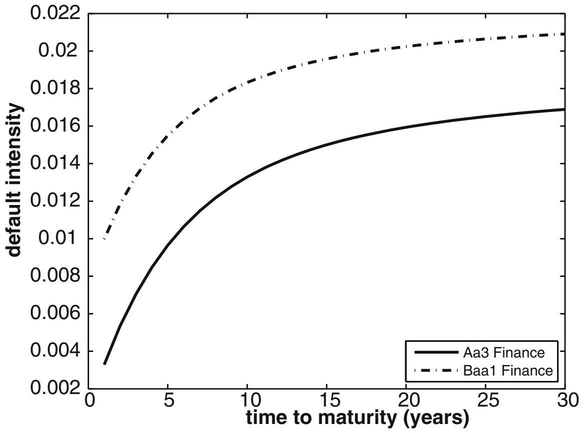


Table 11.12 CDS of two Aa3-rated bonds at $t = 04/30/2010$ (basis points)

$t = 04/30/2010$	Bond 1	Bond 2
1	96.2	62.9
2	112.1	72.9
3	116.3	87.3
4	135.1	105.3
5	141.9	115.1
7	142.4	122.0
10	148.6	126.6

We can see that the CDS of the two securities are different between them; anyway they are not too far from the obtained default-intensity model-based credit spreads, especially over long horizons. The estimated term structures of credit spread can be considered and used as "benchmark" credit spread term structures to price defaultable bonds aggregated for rating and economic sector.

11.5.2 Computational Issues

Stochastic default-risk simulation increases the computational complexity of an ALM system both in terms of amount of data to be managed and computing time.

Specifically, the system has to be able to manage a set of default-risk adjusted term structures, each of them related to a combination of rating and economic sector—the choice of the combinations and the number of the term structures to consider depending on the company investment strategy; the default-intensity parameter computation requires a preprocessing phase implementing a calibration procedure, as, for example, the one described in Sect. 11.5.1. This phase has an impact on the amount of data, needed to properly calibrate the parameters, to be included in the DataBase system; it has also an impact on the execution time required to perform the calibration procedure on each set of data.

Table 11.13 $V(0, Y_T)$, put and call components for three different segregated funds

$N = 5{,}000$	Risk-free bond	Aa3-finance bond	Baa1-finance bond
$V(0, Y_T)$	711.793.017	716.334.673	722.241.384
Std. err.	523.798	535.177	543.725
B_0	690.066.024	697.948.820	705.637.021
P_0	21.726.993	18.385.853	16.604.364
G_0	704.994.119	705.134.076	705.134.076
C_0	6.798.898	11.200.598	17.107.309

After the preprocessing phase, the DataBase must be enriched with all the default-risk parameters calibrated for each default-intensity process; the DataBase Management System has to be able to identify and then to manage credit risky bonds on the basis of the appropriate rating and sector.

In order to analyze the computational overhead due to the introduction of stochastic default-risk modeling in a real case, in this section we again consider the DISAR system. The default risk simulation in DIALMENG, the ALM computing unit of DISAR, requires, for each set of calibrated default-intensity parameters:

1. Simulation of stochastic default-intensity processes
2. Simulation of stochastic default probabilities
3. Computation, at the evaluation date, of the default-risk adjusted term structures

The simulations at points 1 and 2 require, as seen, the use of numerical methods for solving SDEs and Monte Carlo methods. The computation at point 3 requires the evaluation of (11.28), for each considered class.

Each risky bond in the fund has to be managed in order to be "linked" to the pertaining default-risk adjusted term structure and default probabilities, to properly estimate the related financial quantities (value, duration, ...) involved in the considered ALM framework.

We performed numerical simulations considering two different investment portfolios, each composed by the same quantity of just one type of risky bond, Aa3 and Baa1 Finance respectively, with maturity 3 years, fixed annual coupons and same market price at the evaluation date. The policy portfolio contains about two hundreds policies. The time horizon of simulation we consider is 40 years. The SDEs for the risk sources are numerically solved by means of the Euler method with a monthly discretization step. Further, to make some comparisons, we carried out a simulation also on an investment portfolio composed by the same quantity of a risk-free coupon bond, with same maturity and market price.

In Table 11.13 we report the values, in euro, of $V(0, Y_T)$ (and the related standard error) and of the components of the put and call decompositions in (11.6), for the three different segregated funds, obtained performing $N = 5{,}000$ Monte Carlo simulations.

To quantify the increment of computing time overhead, we report in Table 11.14 the execution times of the overall ALM procedure on the investment portfolio

Table 11.14 Execution times (in seconds) for two different segregated funds

N	Risk-free bond	Aa3-finance bond bond	Time increment (%)
6,000	135.812	163.423	20.3
12,000	269	326.333	21.3

Process	$N = 6,000$	$N = 12,000$
1	163.423	326.333
2	85.687	169.048
4	43.234	85.651
6	29.673	59.249
8	22.285	44.494
10	17.899	35.729
12	15.105	30.120

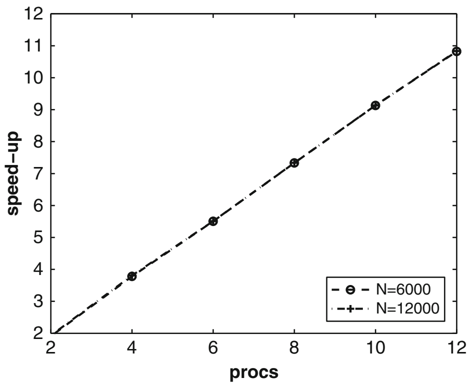


Fig. 11.15 On the *left hand* the execution time (in seconds); on the *right hand* the speedup vs. number of processors

composed by risk-free coupon bonds and on that composed by Aa3-Finance risky bonds (the execution times being the same for the investment portfolio composed by the Baa1-Finance), respectively, for $N = 6{,}000$ and $N = 12{,}000$ MC simulations on one processor. We observe that the default-risk valuation implies an increment of about 20% of the computing time, for both the considered values of N, including just one credit risk class in the investment portfolio.

HPC is therefore mandatory for developing efficient simulation procedures. We implement the same parallelization strategy showed in Sect. 11.4.3. We use the Mersenne-Twister generator included in the *Intel Math Kernel Library* for the generation of pseudo-random sequences.

To analyze the performance of the parallel procedure, we report, on the left hand of Fig. 11.15, the execution times, expressed in seconds, for two values of global number of simulated trajectories, $N = 6{,}000$ and $N = 12{,}000$, versus the number of processors involved in the computation. To evaluate the parallel efficiency, we show, on the right hand of the same figure, the related speedup. The graph reveals the good scalability properties of the algorithm. Indeed, speedup is almost linear. The same behavior was observed in all our experiments.

The parallelization strategy then allows to pull down the execution time, thus allowing to efficiently deal with the complex task of measuring default risk.

11.6 Conclusion

“Risk” and “model” are the key words around which Solvency II project is built. To measure the risk to which an insurance undertaking is exposed, a model has to be defined according to the asset–liability policy; the model must provide market-consistent valuations. The definition of a reliable model depends on robust theoretical basis, accurate numerical approximations, as well as efficient algorithms and high-quality input data. Only high-level technological solutions guarantee high-quality information and perfect timing in risk control and subsequent actions: the Directive requires “measure, monitor, manage and report, on a continuous basis” of the risk.

The Directive allows undertakings to use “internal models,” subject to prior supervisory approval. The implications and requirements introduced by the Directive become particularly compelling when undertakings choose to employ an internal model.

In the case of profit-sharing policies with minimum guarantees, the market-consistent valuation problem clearly shows that “dealing with the time value of money above the classical actuarial technique is nowadays so far off economic reality that it needs to be fundamentally revised” [6]. PS policies are structured contracts which embed options. The structure of these contracts points out the dependence of the company liabilities—and thus of the “future discretionary benefits” in Solvency II jargon—on the segregated fund return; it is therefore clear that the set of rules which form the ALM strategy for the management of the policy portfolio is very complex.

The valuation model has to integrate the dynamic ALM strategy in an MC simulation process of the risk sources. So, time-consuming MC simulations, involving a huge number of random variables on long-time horizons, have to be performed. Efficient numerical algorithms and advanced architectures are thus mandatory for “quickly” providing accurate results.

In this work, some approaches for the solution of the main computational kernels in models for PS policy portfolios valuations have been presented. The analyses we carried out show that robust numerical algorithms implemented in parallel environments allow to obtain effective solution procedures.

The Solvency II Directive is still evolving, thus new computational problems are posed. So, in order to build effective risk-management systems, the focus must be kept on both numerical methods and algorithms for solving computational kernels and different HPC environments (grid architectures, distributed environments, multi-core processors, GPUs) able to provide the necessary computational power.

References

1. G. Barone-Adesi, E. Barone, G. Castagna, Pricing bonds and bond options with default risk. Eur. Financ. Manag. **4**, 231–282 (1998)
2. F. Black, M. Scholes, The pricing of options and corporate liabilities. J. Polit. Econ. **81**(3), 637–654 (1973)
3. D. Brigo, A. Alfonsi, Credit default swaps calibration and option pricing with the SSRD stochastic intensity and interest-rate model. Finance Stochast. **9**(1), 29–42 (2005)
4. D. Brigo, F. Mercurio, *Interest Rate Models: Theory and Practice* (Springer, New York, 2006)
5. S.J. Brown, P.H. Dybvig, The empirical implications of the Cox, Ingersoll, Ross theory of the term structure of interest rates. J. Finance **41**(3), 617–630 (1986)
6. H. Bulhmann, New math for life insurance. Astin Bull. **32**(2), 209–211 (2002)
7. R. Caflisch, W. Morokoff, A. Owen, Valuation of mortgage-backed securities using Brownian bridges to reduce effective dimension. J. Comput. Finance **1**, 27–46 (1998)
8. G. Castellani, L. Passalacqua, Applications of distributed and parallel computing in the solvency II framework: the DISAR system, in *Euro-Par 2010 Parallel Processing Workshops*, ed. by M.R. Guarracino, F. Vivien, J.L. Träff, M. Cannataro, M. Danelutto, A. Hast, F. Perla, A. Knüpfer, B. Di Martino, M. Alexander. Lecture Notes in Computer Science, vol. 6586 (Springer, Berlin, 2011), pp. 413–421
9. G. Castellani, M. De Felice, F. Moriconi, C. Pacati, Embedded Value in Life Insurance. Working paper (2005)
10. G. Castellani, M. De Felice, F. Moriconi, *Manuale di Finanza*, vol. III. Modelli Stocastici e Contratti Derivati (Società editrice il Mulino, Bologna, 2006)
11. S. Corsaro, P.L. De Angelis, Z. Marino, F. Perla, P. Zanetti, On high performance software development for the numerical simulation of life insurance policies, in *Numerical Methods for Finance*, ed. by J. Miller, I.D. Edelman, J. Appleby (Chapman and Hall/CRC, Dublin, 2007), pp. 87–111
12. S. Corsaro, P.L. De Angelis, Z. Marino, F. Perla, P. Zanetti, On parallel asset-liability management in life insurance: A forward risk-neutral approach. Parallel Comput. **36**(7), 390–402 (2010)
13. S. Corsaro, Z. Marino, F. Perla, P. Zanetti, Measuring default risk in a parallel ALM software for life insurance portfolios, in *Euro-Par 2010 Parallel Processing Workshops*, ed. by M.R. Guarracino, F. Vivien, J.L. Träff, M. Cannataro, M. Danelutto, A. Hast, F. Perla, A. Knüpfer, B. Di Martino, M. Alexander. Lecture Notes in Computer Science, vol. 6586 (Springer, Berlin, 2011), pp. 471–478
14. S. Corsaro, P.L. De Angelis, Z. Marino, F. Perla, Participating life insurance policies: An accurate and efficient parallel software for COTS clusters. Comput. Manag. Sci. **8**(3), 219–236 (2011)
15. J.C. Cox, J.E. Ingersoll, S.A Ross. A theory of the term structure of interest rates. Econometrica **53**, 385–407 (1985)
16. G. Deelstra, F. Delbaen, Convergence of discretized stochastic (interest rate) processes with stochastic drift term. Appl. Stochast. Model Data Anal. **14**(1), 77–84 (1998)
17. M. De Felice, F. Moriconi, Market consistent valuation in life insurance: Measuring fair value and embedded options. Giornale dell'Istituto Italiano degli Attuari **67**, 95–117 (2004)
18. M. De Felice, F. Moriconi, Market based tools for managing the life insurance company. Astin Bull. **35**(1), 79–111 (2005)
19. Directive 2009/138/EC of the European Parliament and of the Council of 25 November 2009 on the taking-up and pursuit of the business of Insurance and Reinsurance. Official Journal of the European Union **335**(52), 1–155 (2009)
20. D. Duffie, Credit risk modeling with affine processes. J. Bank. Finance **29**, 2751–2802 (2005)
21. D. Duffie, K.J. Singleton, Modeling term structures of defaultable bonds. Rev. Financ. Stud. **12**(4), 686–720 (1999)

22. D. Duffie, K.J. Singleton, *Credit Risk: Pricing, Measurement, and Management* (Princeton University Press, Princeton, 2003)
23. H. Geman, N. El Karoui, J.C. Rochet. Changes of numéraire, changes of probability measure and option pricing. J. Appl. Probab. **32**(2), 443–458 (1995)
24. P. Glasserman, *Monte Carlo Methods in Financial Engineering* (Springer, New York, 2004)
25. D. Higham, X. Mao, Convergence of Monte Carlo simulations involving the mean-reverting square root process. J. Comput. Finance **8**(3), 35–62 (2005)
26. J. Hull, A. White, One-factor interest rate models and the valuation of interest rate derivative securities. J. Financ. Quant. Anal. **28**, 235–254 (1993)
27. IAIS, Guidance Paper on the Use of Internal Models for Regulatory Capital Purpose, International Association of Insurance Supervisors, Basel, Guidance paper no. 2.6 (2008)
28. J. Jamshidian, An exact bond option formula. J. Finance **44**(1), 205–209 (1989)
29. R.A. Jarrow, D. Lando, S.M. Turnbull, A Markov model for the term structure of credit risk spreads. Rev. Finan. Stud. **10**(2), 481–523 (1997)
30. P.E. Kloeden, E. Platen, *Numerical Solution of Stochastic Differential Equations* (Springer, Berlin, 1992)
31. D. Lando, On Cox processes and credit risky securities. Rev. Derivatives Res. **2**(2–3), 99–120 (1998)
32. D. Lando, *Credit Risk Modeling, Theory and Applications* (Princeton University Press, Princeton, 2004)
33. R. Lord, R. Koekkoek, D. van Dijk, A comparison of biased simulation schemes for stochastic volatility models. Quant. Finance **10**(2), 177–194 (2010)
34. M. Mascagni, A. Srinivasan, Algorithm 806: SPRNG: A scalable library for pseudorandom number generation. ACM Trans. Math. Software **26**, 436–461 (2000)
35. M. Mascagni, S.A. Cuccaro, D.V. Pryor, M.L. Robinson, A fast, high-quality, and reproducible lagged-Fibonacci pseudorandom number generator. J. Comput. Phys. **15**, 211–219 (1995)
36. G.N. Milstein, A method of second-order accuracy integration of stochastic differential equations. Theor. Probab. Appl. **23**, 396–401 (1978)
37. G.N. Milstein, E. Platen, H. Schurz, Balanced implicit methods for stiff stochastic systems. SIAM J. Numer. Anal. **35**(3), 1010–1019 (1998)
38. C. Pacati, Estimating the Euro term structure of interest rates. Research Group on "Models for Mathematical Finance", Working Paper 32 (1999)
39. S. Paskov, J. Traub, Faster valuation of financial derivatives. J. Portfolio Manag. **22**(1), 113–123 (1995)
40. P.J. Schonbucher, *Credit Derivatives Pricing Models – Models, Pricing and Implementation* (Wiley, New York, 2005)

Index