Original Article

A hybrid genetic algorithm–support vector machine approach in the task of forecasting and trading

Received (in revised form): 21st December 2012

Christian L. Dunis

is Emeritus Professor of Banking and Finance at Liverpool John Moores University where he also directed the Centre for International Banking, Economics and Finance (CIBEF) from February 1999 through August 2011. He is a Visiting Professor of Quantitative Finance at the Universities of Venice (Italy) and Aix-en-Provence (France), and at the ECE School of Electronic Engineering in Paris. He is also a consultant to asset management firms, specializing in the application of nonlinear methods to financial management problems.

Spiros D. Likothanassis

is currently Professor and Director of the Pattern Recognition Laboratory, Department of Computer Engineering and Informatics, University of Patras. His research interests include intelligent signal processing and adaptive control, neural networks, genetic algorithms and applications, intelligent agents and applications, bioinformatics, web-based applications, virtual e-learning environments, artificial intelligence/expert systems, data and knowledge mining and intelligent tutoring systems.

Andreas S. Karathanasopoulos

is a senior lecturer in London Metropolitan Business School. In 2008, he received the Master of Science in International Banking and Finance from the Department of Banking and Finance of Liverpool John Moores University. His research interests include financial forecasting, trading strategies, time series prediction, artificial and computational intelligence and neural networks.

Georgios S. Sermpinis

joined the Glasgow Business School in September 2011. He holds degrees from the National Kapodistrian University of Athens and the Liverpool John Moores University. He previously worked at the University of Bedfordshire and Liverpool John Moores University. His research interests include risk management, financial forecasting, trading strategies and artificial intelligence models.

Konstantinos A. Theofilatos

is a PhD candidate in the Department of Computer Engineering and Informatics of the University of Patras, Greece. In 2009, he received a Master's degree from the Department of Computer Engineering and Informatics of the University of Patras. His research interests include computational and artificial intelligence intelligence, evolutionary computation, time series modeling and forecasting, bioinformatics, data mining and web technologies.

Correspondence: Konstantinos A. Theofilatos, Pattern Recognition Laboratory, Department of Computer Engineering & Informatics, University of Patras, Greece
E-mail: theofilk@ceid.upatras.gr

ABSTRACT The motivation of this article is to introduce a novel hybrid Genetic algorithm–Support Vector Machines method when applied to the task of forecasting and trading the daily and weekly returns of the FTSE 100 and ASE 20 indices. This is done by benchmarking its results with a Higher-Order Neural Network, a Naïve Bayesian Classifier, an auto-regressive moving average model, a moving average convergence/divergence model, plus a naïve and a buy and hold strategy. More specifically, the trading performance of all models is investigated in forecast and trading simulations on the FTSE 100 and ASE 20 time series over the period January 2001–May 2010, using the last 18 months for out-of-sample testing.

As it turns out, the proposed hybrid model does remarkably well and outperforms its benchmarks in terms of correct directional change and trading performance.

Journal of Asset Management (2013) **14,** 52–71. doi:10.1057/jam.2013.2;
published online 14 February 2013

Keywords: ASE 20; FTSE 100; trading simulation; genetic algorithms; support vector machines

INTRODUCTION

Forecasting financial time series is a difficult task because of their complexity and their nonlinear, dynamic and noisy behavior. Traditional methods such as autoregressive moving average (ARMA) and moving average convergence/divergence (MACD) models fail to capture the complexity and the nonlinearities that exist in financial time series. On the other hand, nonlinear approaches such as Artificial Neural Networks (ANNs) have given promising empirical evidence but their numerous limitations are often creating skepticism about their use among practitioners (Dunis *et al*, 2009). Support Vector Machines (SVMs) (Vapnik, 2000) handle some of ANNs' limitations as they can be trained more effectively and theoretically provide classification models with enhanced generalization abilities. However, their performance is highly associated with their parameters and input selection that should be selected in a computational manner.

The purpose of this article is to introduce a hybrid Genetic Algorithms (GA) and SVM model, which can overcome some of the limitations of ANNs and simple SVMs and excel in financial forecasting. More specifically, in our hybrid methodology, a GA is used to optimize the SVM parameters and to find the optimal feature subset. Furthermore, the proposed hybrid methodology uses a problem-specific fitness function, which is believed to produce more profitable prediction models.

In our application, we developed a hybrid GA-SVM model and applied it to the task of forecasting and trading the daily and weekly returns of the FTSE 100 and ASE 20 indices. This is done by benchmarking its results with a Higher-Order Neural Network (HONN), a Naïve Bayesian Classifier, an ARMA model, an MACD model, plus a naïve and a buy and hold strategy. More specifically, the performance of all models is investigated in a forecast and trading simulation on the FTSE 100 ASE 20 time series over the period January 2001–May 2010, using the last 18 months for out-of-sample testing.

As it turns out, the proposed hybrid model does remarkably well and outperforms its benchmarks in terms of trading performance. This superiority is also confirmed after transaction costs and leverage to exploit the high information ratios are applied.

The rest of the article is organized as follows. In the next section, we present some relevant recent applications, whereas the subsequent section provides a detailed description of the FTSE 100 and the ASE 20 time series. A detailed overview of the proposed methodology and its benchmarks is given in the latter section, whereas in the penultimate section we present our results. The final section provides some concluding remarks.

LITERATURE REVIEW

The main objective of this article is to introduce a novel hybrid GA and SVM model that can overcome some of the limitations of ANNs and simple SVMs and excel in financial forecasting applications.

Panda and Narasimhan (2007) use a single hidden layer feedforward Neural Network (NN) to produce statistical accurate forecasts

of the INR/USD exchange rate having several linear autoregressive models as benchmarks, whereas Andreou *et al* (2008) use NNs to forecast and trade European options with disappointing results. On the other hand, Kiani and Kastens (2008) forecast the GBP/USD, the CAD/USD and the JPY/USD exchange rates with feedforward and recurrent NNs having as benchmarks several ARMA models. In their application, NNs outperform in statistical terms their ARMA benchmarks in forecasting the GBP/USD and USD/JPY but not in forecasting the USD/CAD exchange rate. Adeodato *et al* (2011) won the NN3 Forecasting Competition problem with an innovative approach based on the use of median for combining MLP forecasts, and Matias and Reboredo (2012) forecast successfully with NNs and other nonlinear models intraday stock market returns. In a forecasting competition, Dunis *et al* (2009 and 2011) and Sermpinis *et al* (2013) compare several Higher-Order NNs and autoregressive models in forecasting and trading the EUR exchange rates. Their results demonstrate the forecasting superiority of a class of NNs, the Psi Sigma, which are able to capture higher-order correlation within their data set.

Until now, many approaches have been based on SVMs for the modeling of financial time series. Cao and Tay (2003) apply SVMs to the problem of forecasting several future contracts from the Chicago Mercantile Market and demonstrate the superiority of SVMs over Back Propagation and regularized Radial Basis Function (RBF) NNs. In the same year, Kim (2003) used SVMs to predict the successful direction of change of the daily Korean composite stock index, whereas Huang *et al* (2005) used SVM to predict correctly the directional movement of the NIKKEI 225 index. More recently, Ince and Trafalis (2008) apply successfully Support Vector Regression to the task of forecasting 10 NASDAQ financial indices.

The proposed model combines GA with SVMs. Lately, some other research groups tried to forecast financial and other time series using algorithmic combinations of GAs and SVMs. In Nguyen *et al* (2009), the authors propose a hybrid methodology that uses a GA to locate the optimal feature subset, which should be used by an SVM classifier. This methodology was applied to financial indices with great success even if the GA was not used to optimize the SVM's parameters and the classification models were not combined with advanced trading strategies. Wu *et al* (2009) developed a novel methodology that used a GA to find the optimal Kernel function and parameters of a Support Vector Regression model. This algorithm was applied to forecast the maximum electrical daily load and outperformed previous models. However, no feature selection procedure was applied in this methodology and it could be improved if the GA search for the optimal features subset on parallel to the Kernel's and parameters' optimization. Min *et al* (2006) introduced a GA to optimize both features subset and the SVM's parameters. Our article extends this methodology by using a novel problem-specific fitness function, by estimating decimal regression values by computing the distance from the classification margin of each sample and by combining the final prediction models with advance trading strategies such as confirmation filters and leverage analysis.

THE FTSE 100 AND ASE20 INDEX

The FTSE 100 index is a share index of the 100 companies listed in the London Stock Exchange with the highest market capitalization. The ASE 20 index consists of the 20 largest Athens Stock Exchange stocks and represents over 50 per cent of the exchange's total capitalization. Both indices are traded by future contracts that are cash settled upon maturity of the contract with the value of the index fluctuating on a daily basis.

Table 1: NN and SVM–GA data sets

Name of period	*Beginning*	*End*
Total data set	1 January 2001	31 May 2010
Training data set	1 January 2001	6 November 2006
Test data set	7 November 2006	11 August 2008
Validation set	12 August 2008	31 May 2010

The cash settlement of this index is simply determined by calculating the difference between the traded price and the closing price of the index on the expiration day of the contract. Both series were provided by Bloomberg's financial information services.

The FTSE 100 and ASE 20 daily and weekly time series are non-normal (Jarque-Bera statistics confirms this at the 99 per cent confidence interval), containing slight skewness and high kurtosis. They are also nonstationary and we decided to transform the series into stationary series of daily and weekly rates of return.[1]

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

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

The summary statistics of the FTSE 100 and ASE 20 daily and weekly returns series reveal positive skewness and high kurtosis. The Jarque-Bera statistic confirms again that the return series are non-normal at the 99 per cent confidence level. These return series will be forecasted from our models.

For each time series under study, as inputs to our algorithms and our networks, we selected the first 14 autoregressive lags of the series. In order to train our artificial intelligence models, we further divided our data set as shown in Table 1.

FORECASTING MODELS

Benchmark models

In this article, we benchmark our proposed methodology with four traditional strategies, namely, an ARMA, an MACD technical model, the buy and hold strategy and a naïve strategy, plus the Naïve Bayesian Classifier and an HONN.

Buy and hold strategy

This is a simple strategy, where we buy the index (asset) at the beginning of the review period and sell it back at the end.

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:

$$\hat{Y}_{t+1} = Y_t \qquad (2)$$

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

Moving average

The moving average model is defined as:

$$M_t = \frac{(Y_t + Y_{t-1} + Y_{t-2} + \cdots + Y_{t-n+1})}{n} \qquad (3)$$

where M_t is the moving average at time t; n is the number of terms in the moving average; 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.

The forecaster must use judgment 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 models selected were a combination of (1,3) for FTSE 100 daily, (2,9) for the FTSE 100 weekly, (1,7) for the ASE 20 daily and (1,3) for the ASE 20 weekly series.

ARMA model

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).[2]

The ARMA model takes the form:

$$\begin{aligned} Y_t =& \phi_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots \\ &+ \phi_p Y_{t-p} + \varepsilon_t - w_1 \varepsilon_{t-1} \\ &- w_2 \varepsilon_{t-2} - \cdots - w_q \varepsilon_{t-q} \end{aligned} \quad (4)$$

where Y_t is the dependent variable at time t; Y_{t-j}, $j = 1 \ldots p$ are the lagged dependent variable; ϕ_j, $j = 1 \ldots p$ are regression coefficients; ε_t is the residual term; ε_{t-m}, $m = 1 \ldots q$ are previous values of the residual; w_m, $m = 1 \ldots q$ are weights.

Using the correlogram and the information criteria in the training and the test sub-periods as a guide, we choose our ARMA models for the four series under study (for more information see Tables A1-A4). All of their coefficients are significant at the 99 per cent confidence interval. The selected ARMA models for the daily FTSE 100, weekly FTSE 100, daily ASE 20 and weekly ASE 20 series are presented in equations (5)-(8), respectively:

$$\begin{aligned} Y_t =& -0.00008 - 1.04212 Y_{t-1} \\ &- 0.90703 Y_{t-2} - 0.45808 Y_{t-3} \\ &+ 0.98777 \varepsilon_{t-1} + 0.79846 \varepsilon_{t-2} \\ &+ 0.282869 \varepsilon_{t-3} \end{aligned} \quad (5)$$

$$\begin{aligned} Y_t =& -0.00006 - 1.44551 Y_{t-1} \\ &- 0.59617 Y_{t-2} + 0.39824 Y_{t-4} \\ &+ 0.43422 Y_{t-5} + 1.49402 \varepsilon_{t-1} \\ &+ 0.71058 \varepsilon_{t-2} - 0.60923 \varepsilon_{t-4} \\ &- 0.64361 \varepsilon_{t-5} \end{aligned} \quad (6)$$

$$\begin{aligned} Y_t =& -0.00009 + 0.38488 Y_{t-1} - 0.92571 Y_{t-2} \\ &- 0.37892 \varepsilon_{t-1} + 0.92536 \varepsilon_{t-2} \end{aligned} \quad (7)$$

$$\begin{aligned} Y_t =& -0.00010 + 1.36291 Y_{t-1} - 1.29897 Y_{t-2} \\ &+ 0.51318 Y_{t-3} - 1.38099 \varepsilon_{t-1} + 1.364626 \varepsilon_{t-2} \\ &- 0.41627 \varepsilon_{t-3} \end{aligned} \quad (8)$$

The models selected are retained for out-of-sample estimation. The performance of the strategy is evaluated in terms of trading performance.

The HONN architecture

NNs exist in several forms in the literature. The most popular architecture is the Multi-Layer Perceptron (MLP). A standard NN 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 layers 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 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[3] (Shapiro, 2000). The learning algorithm simply tries to find those weights, which minimize an error function (normally the sum of all squared differences between target and actual values). As 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

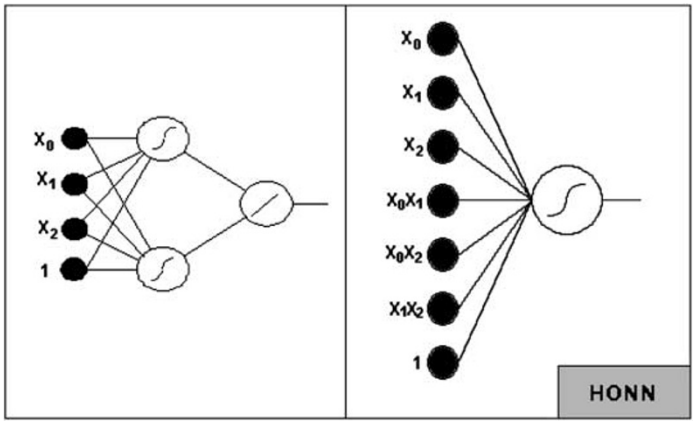

Figure 1: Left, MLP with three inputs and two hidden nodes; right, second-order HONN with three inputs.

time to prevent overfitting (this is called 'early stopping'). This can 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 optimized by maximizing the forecasting accuracy for the test data set. Then the predictive value of the model is evaluated by applying it to the validation data set (out-of-sample data set).

HONNs were first introduced by Giles and Maxwell (1987) and were called 'Tensor Networks'. Although the extent of their use in finance has so far been limited, Knowles *et al* (2011) show that, with shorter computational times and limited input variables, 'the best HONN models show a profit increase over the MLP of around 8 per cent' on the EUR/USD time series (p. 7). 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 nonlinear data, and consequently provide superior simulations compared with those produced by ANNs' (p. 188). Furthermore, HONNs clearly outperform in terms of annualized return and this enables Dunis *et al* (2008) to conclude with confidence over their forecasting superiority and their stability and robustness through time.

Although they have already experienced some success in the field of pattern recognition and associative recall,[4] HONNs have only started recently to be used in finance. The architecture of a three-input second-order HONN is shown in Figure 1.

Where $x_t^{[n]}(n = 1,2, \ldots, k+1)$ 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; ● are the model inputs.; ⊘ is the transfer sigmoid function:

$$S(x) = \frac{1}{1+e^{-x}} \tag{9}$$

⊘ is a linear function:

$$F(x) = \sum_i x_i. \tag{10}$$

The error function to be minimized is:

$$E(u_{jk}, w_j) = \frac{1}{T}\sum_{t=1}^{T}\left(y_t - \tilde{y}_t(u_{jk},)\right)^2, \quad \text{with } y_t \text{ being the target value} \tag{11}$$

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 four 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 NNs can be largely avoided. For a complete description of HONNs, see Knowles *et al* (2011, p. 52).

The parameters that were applied for the HONNs deployed in this article are presented in detail in Table C2.

Naïve Bayesian methodology

Recent work in supervised learning has shown that a surprisingly simple Bayesian classifier with strong assumptions of independence among features, called naive Bayes, is competitive with state-of-the-art classifiers such as SVM, random forests and so on. Naïve Bayes classifier (Howson and Urbach, 1993) is a simple probabilistic classifier based on applying the Bayes theorem. Bayes' theorem expresses the conditional probability, or 'posterior probability', of a hypothesis H (that is, its probability after evidence E is observed) in terms of the 'prior probability' of H, the prior probability of E and the conditional probability of E given H. It implies that evidence has a stronger confirming effect if it was more unlikely before being observed. Bayes' theorem is valid in all common interpretations of probability and it is commonly applied in science and engineering.

In this article, we applied the naïve Bayesian methodology in the problem of predicting the movement direction of the ASE-20 Greek Stock index using as features-inputs only previous values of the index's returns. The problem of predicting the movement direction of the ASE-20 Greek Stock index is simply classification problem with two classes representing positive or negative movement.

Bayesian-based classifiers are the most common computational methods used to integrate data from a wide variety of sources, including both experimental results, functional, structural and sequential information. They use these features to assess the likelihood that a particular potential protein interaction is a true positive result. Bayesian-based classifiers are a widely used class of probabilistic methods known for their high ability to achieve interpretable classification. Using Bayesian-based classifiers for the problem of predicting PPIs, in addition to a high-quality classification between existing and nonexisting interactions, researchers are able to obtain a confidence score for every interaction in a straightforward manner from the probabilities produced by the method.

Bayesian classifiers, when applied in the movement direction of ASE-20 Greek Stock index, classify the next day movement direction in positive if the inequality (12) holds. Otherwise, the next day movement direction is predicted as negative.

$$\begin{aligned} &P(positive|a_1, a_2, \ldots, a_n) \\ &\quad > P(negative|a_1, a_2, \ldots, a_n) \\ &P(\text{true interaction}|a_1, a_2, \ldots, a_n) \\ &\quad > P(\text{false interaction}|a_1, a_2, \ldots, a_n) \end{aligned} \quad (12)$$

where $a_1, a_2, \ldots, a_n$ are the values of the inputs that we used.

Using Bayes' theorem, the inequality (12) can be rewritten as:

$$\begin{aligned} &\frac{P(a_1, a_2, \ldots, a_n|positive) * P(positive)}{P(a_1, a_2, \ldots, a_n)} \\ &> \frac{P(a_1, a_2, \ldots, a_n|negative) * P(negative)}{P(a_1, a_2, \ldots, a_n)} \end{aligned} \quad (13)$$

Simplifying the term $P(a_1, a_2, \ldots, a_n)$, we get the following classification condition:

$$P(a_1, a_2, \ldots, a_n|positive) * P \times (positive) > P \times (a_1, a_2, \ldots, a_n|negative) * P \times (negative) \quad (14)$$

The probabilities P(positive) and P(negative) are estimated from the in-sample data set. Estimating effectively the probabilities $P(a_1, a_2, \ldots, a_n|\,\text{positive})$ and $P(a_1, a_2, \ldots, a_n)$ is computationally very hard and requires a huge sample size. This is the main disadvantage of the full Bayesian approaches (Howson and Urbach, 1993).

To overcome the latter disadvantage, naïve Bayesian method, which is a simplification of the full Bayesian approach, was developed. Naïve Bayesian method assumes independence between the features and computes probabilities of the inequality (14) in the following manner:

$$P(a_1, a_2, \ldots, a_n|positive) = \prod_{i=1}^{n} P(a_i|positive) \quad (15)$$

$$P(a_1, a_2, \ldots, a_n|negative) = \prod_{i=1}^{n} P(a_i|negative) \quad (16)$$

The probabilities $P(a_i|\,\text{positive})$ and $P(a_i|\,\text{negative})$ are estimated from the in-sample data.

Among the many possible machine-learning approaches that could be applied to classification problems, Naïve Bayesian methodology presents the following advantages:

- They allow for combining highly dissimilar types of data, converting them to a common probabilistic framework, without unnecessary simplification.
- In contrast to 'black box' approaches, Bayesian networks are readily interpretable as they represent conditional relationships among information sources.

For the present work, in order to implement the Naïve Bayesian approach for the problem of predicting the daily and weekly returns of the FTSE 100 and ASE 20 series, we used the Statistical toolbox of Matlab R2010b edition. The characteristics of the Naïve Bayesian Classifier approach that were used in this article are presented in Table D1.

Support vector machines

SVMs are a group of supervised learning methods that can be applied to classification or regression. SVMs represent an extension to nonlinear models of the generalized algorithm developed by Vapnik (2000). They have developed into a very active research area and have already been applied to many scientific problems. Specifically, SVMs have already been applied in many prediction and classification problems in finance and economics (Cao and Tay, 2003; Kim, 2003; Huang *et al*, 2005 and Ince and Trafalis, 2008), although they are still far from mainstream, and the few financial applications so far have only been published in statistical learning and artificial intelligence journals.

SVM models were originally defined for the classification of linearly separable classes of objects. For any particular linear separable set of two-class objects, SVMs are able to find the optimal hyperplanes that separates them providing the bigger margin area between the two hyperplanes. The mathematical explanation of this ability is described in the next sub-section.

SVMs can also be used to separate classes that cannot be separated with a linear classifier. In such cases, the coordinates of the objects are mapped into a feature space using nonlinear functions. The feature space in which every object is projected is a high-dimensional space in which the two classes can be separated with a linear classifier. This procedure is explained mathematically in the subsequent sub-section.

Linear separability of data and linear SVMs

Suppose we are given a set of examples $(x_1, y_1), \ldots, (x_l, y_l)$, where $x_i \in R^N$ and $y_i \in (\pm 1)$ are the input patterns and their class labels, respectively. In this section, we assume that the two classes of the classification problem are linearly separable (this is not usually the case in real data). In this case, we can find an optimal weight vector w_0 such that $||w_0||^2$ is minimum (to maximize the margin $\Delta = 2/||w_0||$ of separation (Scholkopf *et al*, 1999) and $y_i^*(w_0^* x_i + b_0) \geqslant 1$, $i = 1, \ldots, l$.

The support vectors are the training examples x_i that satisfy the equality, that is, $y_i^*(w_0^* x_i + b_0) = 1$ and they define two hyperplanes. The one hyperplane goes through the support vectors of one class and the other through the support vectors of the other class. The distance between the two hyperplanes is maximized when the norm of the weight vector $||w_0||^2$ is minimum. This minimization can be realized by maximizing the following function with respect to the variables α_i (Lagrange multipliers) (Vapnik, 2000):

$$W(\alpha) = \sum_{i=2}^{l} \alpha_i - \frac{1}{2\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i * \alpha_j * y_i * y_j * < x_i, x_j >} \quad (17)$$

subject to the constraints: $0 \leqslant \alpha_i$ and $\sum_{i=1}^{l} \alpha_i * y_i = 0$. If $\alpha_i > 0$, then x_i corresponds to a support vector. The classification of an unknown vector $\mathbf{x}$ is obtained by computing:

$$F(x) = \text{sgn}\{w_0 * x + b_0\} \text{ where } w_0 = \sum_{i=1}^{l} \alpha_i * y_i * x_i \quad (18)$$

and the sum only takes into account $N_S \leqslant 1$ nonzero support vectors (that is, training set vectors x_i whose α_i are nonzero). Clearly, after the training, the classification can be accomplished efficiently by taking the dot product of the optimum weight vector w_0 with the input vector $\mathbf{x}$. Simple MLP Neural Networks estimate a simple hyperplane to separate the two classes in the feature space. SVMs not only estimate such a hyperplane, but they also reassure that the estimated hyperplane is the optimal one as it maximizes the margin of separation achieving better generalization properties. Thus, SVMs are considered superior classifiers mathematically compared with other forms of NNs.

Nonlinear separability of data and nonlinear SVMs

The cases in which the data are not linearly separable, as in financial modeling problems, are handled by introducing slack variables $(\xi_1, \xi_2, \ldots, \xi_l)$ with $\xi_1 \geqslant 0$ such that $y_i^*(w^* x_i + b_0) \geqslant 1 - \xi_i$, $i = 1, \ldots, l$. The introduction of the variables ξ_i allows misclassified points, which have their corresponding $\xi_I > 1$. Thus, $\sum_{i=1}^{l} \xi_i$ is an upper bound on the number of training errors. The corresponding generalization of the concept of optimal separating hyperplane is obtained by the solution of the following optimization problem:

$$\text{Minimize } \frac{1}{2} w * w + C * \sum_{i=1}^{l} \xi_i \quad (19)$$

subject to

$$y_i * (w * x_i + b_0) 1 - \xi_i \text{ and } \xi_i 0, i = 1, ..., l \quad (20)$$

The control of the learning capacity is achieved by the minimization of the first part of equation (16), whereas the purpose of the second term is to punish for misclassification errors. The parameter C is a kind of regularization parameter that controls the tradeoff between learning capacity and training set errors. Clearly, a large C corresponds to assigning a higher penalty to training errors and thus increasing training performance. However, extremely large C values may lead to overfitted classification

models with decreased generalization abilities.

Finally, the case of nonlinear SVMs should be considered. The input data in this case are mapped into a high-dimensional feature space through some nonlinear mapping Φ chosen *a priori* (Cortes and Vapnik, 1995 and Scholkopf *et al*, 1999). The optimal separating hyperplanes are then constructed in this space.

The corresponding optimization problem is obtained from equation (14) by substituting x by its mapping $z = \Phi(x)$ in the feature space, that is, the maximization of $W(\alpha)$. In addition, the constraint $0 \leqslant \alpha_i$ becomes $0 \leqslant \alpha_i \leqslant C$ (assuming the nonseparable case). When it is possible to derive a proper kernel functional K such that $K(x_i, x_j) = <\Phi(x_i), \Phi(x_j)>$, the mapping Φ is not explicitly used. Conversely, given a symmetric positive kernel $K(x, y)$, Mercer's theorem (Scholkopf and Smola, 2002) states that there exists a mapping Φ such that $K(x, y) = <\Phi(x_i), \Phi(x_j)>$. By designing a kernel K that satisfies Mercer's condition, the training algorithm is reformulated to the maximization of

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \times \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i * \alpha_j * K(x_i, x_j) * y_i * y_j \tag{21}$$

with the constraints $0 \leqslant \alpha_i \leqslant C$, and $\sum_{i=0}^{l} \alpha_i * y_i = 0$ and the decision function becomes

$$F(x) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i * y_i * K(x, x_i) + b_0 \right) \tag{22}$$

With different expressions for inner products $K(x, x_i)$, we can construct different learning machines with arbitrary types of decision surfaces (nonlinear in input space). The best known kernel types are the polynomial and the radial basis. Polynomial kernels specify polynomials of any fixed order d for the inner product in the corresponding feature space, that is,

$$K(x, x_i) - ((x_i * x) + 1)^d \tag{23}$$

and the RBF kernel has the form

$$K(x, x_i) * \exp(-\gamma * \|x - x_i\|^2) \tag{24}$$

The RBF kernels construct decision functions of the form:

$$F(x) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i * y_i * \exp(-\gamma * \|x - x_i\|^2 + b_0) \right) \tag{25}$$

In the case of the RBF kernel type, the SVM training algorithm determines the centers (support vectors) x_i, the corresponding weights α_i and the threshold b_0. This kernel nonlinearly maps samples into a higher dimensional space so that it, unlike the linear kernel, can handle the case when the relation between class labels and attributes is nonlinear. In our approach, we used the RBF kernel because of its higher reliability in finding optimal classification solutions in most practical situations (Kerthi and Lin (2003). The second reason is the number of hyperparameters that influences the complexity of model selection. The polynomial kernel has more hyperparameters than the RBF kernel and thus it needs a more complex procedure for its parameter optimization procedure. The parameter of the RBF kernel that should be optimized is gamma, which is included in equations (24) and (25).

Proposed methodology

In this section, we describe the proposed methodology. The proposed methodology is a hybrid method of GAs and SVMs specialized for trading financial assets.

When using SVMs, two major decisions must be made. The feature subset used as input to the classifier and the SVM parameters must be optimized. In order to optimize both, we used GAs that are heuristic

evolutionary techniques known for their potential in hard optimization problems.

GAs (Holland, 1995) are search algorithms inspired by the principle of natural selection. They are useful and efficient if the search space is big and complicated or there is not any available mathematical analysis of the problem. A population of candidate solutions, called *chromosomes*, is optimized via a number of evolutionary cycles and genetic operations, such as *crossovers* or *mutations*. Chromosomes consist of genes, which are the optimizing parameters. At each iteration (*generation*), a fitness function is used to evaluate each chromosome, measuring the quality of the corresponding solution and the fittest chromosomes are selected to survive. This evolutionary process is continued until some termination criteria are met. It has been shown that GAs can deal with large search spaces and do not get trapped in local optimal solutions like other search algorithms (Holland, 1995).

In our approach, we use a simple GA where each chromosome comprises *feature genes* that encode the best feature subset and *parameter genes* that encode the best choice of parameters. The parameters that are optimized using GAs are the parameters C and gamma used by SVMs. As described in the previous section, the parameter C is a kind of regularization parameter that controls the tradeoff between learning capacity, training set errors and generalization ability, and gamma is a parameter of the RBF kernel function.

For the GA used in our hybrid methodology, the one-point crossover and the mutation operators were used. One-point crossover creates two offsprings from every two parents. The parents are selected at random, a crossover point c_x is selected at random and two offsprings are made by both concatenating the genes that precede c_x in the first parent with those that follow (and include) c_x in the second parent. The probability for selecting an individual as a parent for the crossover operator to be applied is named as crossover probability. The offspring produced by the crossover operator replace their parents in the population. The mutation operator places random values in randomly selected genes with a certain probability named as mutation probability. Mutation operator is very important for avoiding local optima and exploring a larger surface of the search space. Crossover and mutation probabilities for the GA were set to 0.9 and 0.1, respectively. Crossover is used in hope that new chromosomes will have good parts of old chromosomes and maybe the new chromosomes will be better. However, it is good to leave some part of population survive to next generation. This is the reason a high (but not equal to one) crossover probability was used. As already mentioned, mutation is made to prevent falling GA into local extreme, but it should not occur very often, because then GA will in fact change to random search. That is the main reason why a small mutation probability was applied.

For the selection step of the GA, roulette selection (Holland, 1995) was used. In roulette selection, chromosomes are selected according to their fitness. The better the chromosomes are, the more chances they have to be selected. In our approach, elitism was used to raise the evolutionary pressure in better solutions and to accelerate the evolution. By using elitism, we assured that the best solution is copied without changes to the new population; hence, the best solution found can survive at the end of every generation. The fitness function is defined as in equation (26):

$$fitness = accuracy + accumulated\_return \quad (26)$$

where *accuracy* is the SVM accuracy in the in-sample test set and *accumulated_return* is the accumulated return of the SVM in the sample test set. We chose this fitness function to balance the accuracy and financial effectiveness of the classifiers. The size of the initial population was set to 30 chromosomes and the termination criterion is the maximum number of 50 generations to be

reached, combined with a termination method that stops the evolution when the population is deemed as converged. The population is deemed as converged when the average fitness across the current population is less than 5 per cent away from the best fitness of the current population. Specifically, when the average fitness across the current population is less than 5 per cent away from the best fitness of the population, the diversity of the population is very low and evolving it for more generations is unlikely to produce different and better individuals than the existing ones or the ones already examined by the algorithm in previous generations. The flowchart of the proposed methodology is depicted in detail in Figure 2.

The inputs that are selected in the best execution of the proposed methodology are the 1, 2, 4, 5, 6, 12 and 13 autoregressive lag for the ASE 20 daily return series and the 1, 3, 4, 6, 7, 8 and 10 autoregressive lag for the ASE 20 weekly series. For the FTSE 100 daily and weekly series, our algorithm selected the 1, 2, 3, 5, 7, 12 and 1, 3, 4, 5, 6, 8, 10, 12 autoregressive lag of the series under study, respectively. The parameter C and gamma were set by the hybrid proposed methodology to 4.14 and 12.99, 4.55 and 14.01, 4.98 and 13.11, 4.01 and 12.55 for the ASE 20 daily, ASE 20 weekly, FTSE 100 daily and FTSE 100 weekly series, respectively. This comparatively small values for the parameter C forces the model not to overfit in the training data and thus to enhance its performance over the out-of-sample data set. A detailed list of the parameters used for the proposed methodology is presented in Table C1.

EMPIRICAL TRADING SIMULATION RESULTS

In this section, we present the results of the proposed methodology applied to trading the FTSE 100 and ASE 20 stock indices. These results (the performance metrics are described in detail in Table B1) are compared with the results of the retained benchmark models. The trading strategy applied in this article is to go or stay 'long' when the forecast return is above zero and go or stay 'short' when the forecast return is below zero. When the forecast return is 0, we keep our position.

According to the Athens Stock Exchange, transaction costs for financial institutions and fund managers dealing with a minimum of 143 contracts or 1 million euros is 10 euros per contract (round trip). Dividing this transaction cost of the 143 contracts by the average size deal (1 million euros) gives us an average transaction cost for large players of 14 basis points or 0.14 per cent per position. Following a similar approach, the transaction costs for the FTSE 100 contracts are assumed to be 0.06 per cent per position. Table 2 presents the trading performance of our models.

We can see that the proposed methodology performs significantly better than the other benchmark methods in terms of correct directional change, information ratio and annualized return for all series under study. HONNs and the Naïve Bayesian Classifier present the second and the third best performance, respectively. From our statistical and technical benchmarks, only the MACD for all series and the Naïve for the ASE 20 daily series present positive annualized returns.

Leverage to exploit high information ratios

To further improve the trading performance of our models, we introduce a 'level of confidence' to our forecasts, that is, a leverage based on the test sub-period. The leverage factors applied are calculated in such a way that each model has a common volatility equal to the average annualized volatility of each series on the test data set. This leverage was applied to all models that achieved an in-sample information ratio of at least 2 and, as such, would have been candidates for leveraging out-of-sample.

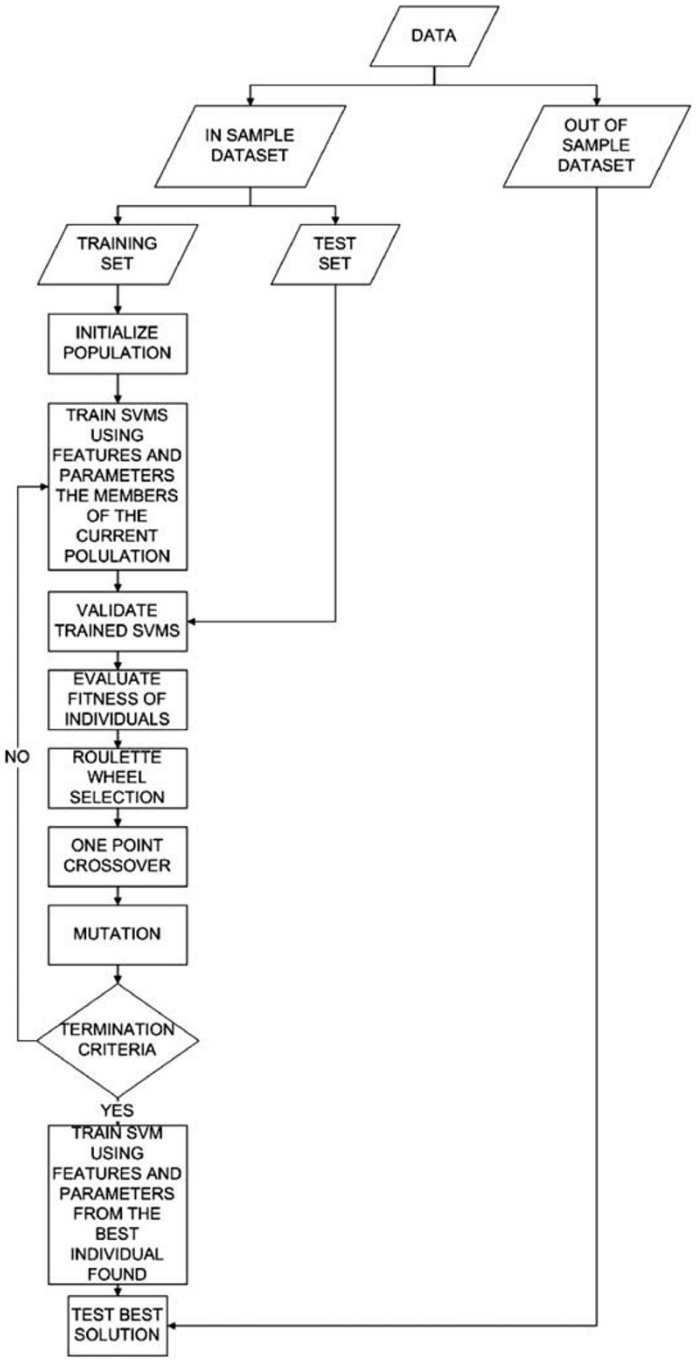

Figure 2: Flowchart of proposed methodology.

The transaction costs are calculated by taking 0.14 per cent per position into account, whereas the cost of leverage (interest payments for the additional capital) is calculated at 4 per cent p.a. (that is 0.016 per cent per trading day).[5] Our results are presented in Table 3.

From Table 3, we note that all our models manage to exploit the leverage applied and to present higher annualized returns, despite the leverage costs. Moreover, we note that the ranking of our best models is retained.

Table 2: Out-of-sample trading performance results

	Buy and hold	*NAIVE*	*MACD*	*ARMA*	*Naïve Bayesian classifier*	*HONNs*	*Proposed methodology*
ASE 20 daily							
Information ratio	−0.87	0.70	0.00	−0.09	0.70	0.85	0.98
Annualized return (including costs)	−38.16%	30.98%	0.09%	−4.26%	31.13%	37.77%	43.58%
Maximum drawdown	−87.24%	−32.27%	−46.26%	−46.92%	−29.61%	−31.72%	−27.34%
Correct directional change	45.11%	47.02%	47.45%	48.13%	51.91%	50.21%	53.83%
ASE 20 weekly							
Information ratio	−0.38	−0.93	0.41	−0.11	0.97	1.21	1.38
Annualized return (including costs)	−7.04%	−17.35%	7.64%	−2.03%	15.73%	18.74%	25.24%
Maximum drawdown	−31.49%	−35.10%	−13.68%	−16.83%	−16.91%	−15.91%	−13.69%
Correct directional change	47.01%	45.97%	48.92%	48.17%	52.19%	54.13%	57.33%
FTSE 100 daily							
Information ratio	0.49	−1.52	0.30	0.11	1.03	1.54	2.15
Annualized return (including costs)	−4.85%	−14.82%	2.88%	1.08%	9.11%	14.85%	20.68%
Maximum drawdown	−11.79%	−24.49%	−17.48%	−7.32%	−6.59%	−5.93%	−4.37%
Correct directional change	52.08%	47.99%	51.74%	48.22%	51.87%	52.45%	53.62%
FTSE 100 weekly							
Information ratio	−0.22	−0.83	0.23	−0.51	0.64	0.97	1.20
Annualized return (including costs)	−4.16%	−15.54%	4.45%	−9.75%	12.03%	17.90%	22.23%
Maximum drawdown	−31.49%	−35.10%	−23.68%	−27.50%	−23.71%	−19.31%	−17.15%
Correct directional change	48.20%	46.67%	49.33%	46.82%	52.17%	52.86%	56.00%

Table 3: Out-of-sample trading performance – Leverage

	Naïve Bayesian classifier	*HONNs*	*Proposed methodology*
ASE 20 daily			
Information ratio	0.70	0.85	0.98
Annualized return (including costs)	34.86%	40.30%	50.70%
Maximum drawdown	−32.28%	−33.63%	−30.89%
Leverage factor	1.09	1.06	1.13
ASE 20 weekly			
Information ratio	0.97	1.21	1.38
Annualized return (including costs)	16.97%	19.12%	29.02%
Maximum drawdown	−19.10%	−15.91%	−13.69%
Leverage factor	1.10	1.04	1.16
FTSE 100 daily			
Information ratio	1.03	1.54	2.15
Annualized return (including costs)	9.24%	15.72%	21.97%
Maximum drawdown	−9.01%	−7.13%	−6.59%
Leverage factor	1.07	1.12	1.09
FTSE 100 weekly			
Information ratio	0.64	0.97	1.20
Annualized return (including costs)	12.72%	19.12%	22.84%
Maximum drawdown	−26.09%	−21.15%	−19.92%
Leverage factor	1.10	1.13	1.05

Confirmation filters

Up to now, the trading strategies applied to the models use a zero threshold: they suggest to go long when the forecast is above zero and to go short when the forecast is below zero. In the following, we examine how the models behave if we introduce a threshold d around zero (see Figure 3) and what happens if we vary that threshold.

The filter rule for all our models is presented in Figure 3.

An optimistic investor will prefer a threshold near to 0, whereas a pessimistic investor will prefer to invest only when the signal from his model is strong and will prefer a threshold of more than 2 per cent. Because of its nature, our buy and hold strategy is not applicable for our confirmation filters. The proposed methodology's outputs are in binary form predicting the movement of the examined index one day ahead. To apply the confirmation filters methodology, we used the distances from the classification margins of the trained SVM models to compute the regression prediction values.

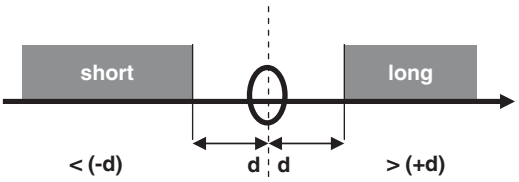

Figure 3: Filtered trading strategy with one single parameter.

In Table 4, we present the performance of our models for thresholds 0 to 2.5 per cent. The entries represent the annualized return values.

From Table 4, we note for most models under study a threshold of 0 providing the higher annualized return. It is also interesting to note that the perfect threshold in the out-of-sample period often coincides with the best threshold in the in-sample period. A conservative investor, who will prefer a threshold of above 2 per cent, will see his profits diminishing with the notable exception of Naïve strategy for the FTSE 100 daily series. On the other hand, an optimistic investor will be rewarded for his confidence over his models with higher annualized returns.

Table 4: Out-of-sample trading performance – Confirmation filters

	Thresholds	*NAIVE*	*MACD*	*ARMA*	*Naïve Bayesian classifier*	*HONNs*	*Proposed methodology*
ASE 20 daily	*0*	30.98*	**0.09**	**−4.26***	**31.13***	**37.77***	43.58
	0.5	32.11	−3.15*	−18.44	18.91	30.95	**49.17***
	1	30.05	−2.91	−5.57	15.19	21.11	28.71
	1.5	**34.65**	−7.17	−10.84	13.00	13.87	4.04
	2	32.37	−3.79	−6.17	9.81	12.99	6.08
	2.5	29.56	−4.19	−6.17	10.12	10.97	10.03
ASE 20 weekly	0	**−17.35**	**7.64***	**−2.03***	15.73*	**18.74***	**25.24***
	0.5	−19.47*	4.15	−2.39	**17.37**	15.71	24.11
	1	−21.18	1.67	−2.16	13.81	12.86	20.84
	1.5	−18.58	−3.85	−2.47	12.85	17.95	17.69
	2	−19.51	−8.17	−2.80	10.75	10.68	15.91
	2.5	−22.18	−6.18	−3.48	11.84	11.11	13.02
FTSE 100 daily	0	−14.82	**2.88***	**1.08***	9.11	14.85	**20.68***
	0.5	−10.41*	0.06	−2.94	**12.15***	18.57*	10.42
	1	−8.61	−5.81	−4.91	10.53	**19.68**	10.18
	1.5	−5.33	−7.16	−5.85	9.61	10.75	10.04
	2	**−3.59**	−10.57	−6.95	6.87	6.38	10.04
	2.5	−4.22	−14.68	−4.72	5.95	5.82	10.35
FTSE 100 weekly	0	−15.54*	**4.45***	**−9.75***	12.03	17.90*	**22.23***
	0.5	**−2.59**	3.10	−10.94	**14.89***	**19.57**	21.59
	1	−5.19	2.51	−12.50	10.96	15.47	17.48
	1.5	−3.50	2.96	−18.49	11.57	14.62	19.58
	2	−10.41	2.47	−15.38	12.07	13.53	8.85
	2.5	−15.97	2.01	−16.69	7.38	10.73	12.96

Note: The entries in bold represent the best confirmation filter in the out-of-sample period while the entries in * represent the best confirmation filter in the in-sample period.

CONCLUDING REMARKS

In this article, a novel hybrid GA-SVM method was introduced for the task of forecasting and trading the daily and weekly returns of the FTSE 100 and ASE 20 indices. The proposed method's results were benchmarked with an HONN, a Naïve Bayesian Classifier, an ARMA, an MACD, plus a naïve and a buy and hold strategy. More specifically, the trading and statistical performance of all models were investigated in a forecast and trading simulation on the daily and weekly FTSE 100 and ASE 20 time series over the period January 2001–May 2010 using the last 18 months for out-of-sample testing. Only autoregressive terms were used as inputs for all the forecasting models.

In terms of our results, the proposed methodology outperforms all its benchmarks in terms of correct directional change, annualized return and information ratio after transaction costs. This superiority is confirmed when a simple leverage is applied to our best models. Moreover, after the application of confirmation filters, we can argue that an optimistic investor with confidence over the forecasting ability of his models would be able to enjoy a higher trading performance on the time series and period under study.

Finally, our experimental results could be a step toward convincing a growing number of quantitative fund managers to experiment beyond the bounds of traditional statistical and NN models. The proposed hybrid methodology should be applied for the task of forecasting and trading the directional movement of many other indexes in order to prove its ability to extract reliable and trustworthy prediction and trading models.

NOTES

1. The percentage return is linearly additive but the log return is not linearly additive across portfolio components.

2. For a full discussion on the procedure, refer to Box *et al* (1994).
3. Backpropagation networks are the most common multi-layer networks and are the most commonly used type in financial time series forecasting (Kaastra and Boyd, 1996).
4. Associative recall is the act of associating two seemingly unrelated entities, such as smell and color.
5. The interest costs are calculated by considering a 4 per cent 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. However, 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.

REFERENCES

Adeodato, P., Arnaud, A., Vasconcelos, G., Cunha, R. and Monteiro, D. (2011) MLP ensembles improve long term prediction accuracy over single networks. *International Journal of Forecasting* 27(3): 661–671.

Andreou, P., Charalampous, C. and Martzoukos, S. (2008) Pricing and trading European options by combining artificial neural networks and parametric models with implied parameters. *European Journal of Operational Research* 185(3): 1415–1433.

Box, G., Jenkins, G. and Gregory, G. (1994) *Time Series Analysis: Forecasting and Control*. Hoboken, NJ: Prentice-Hall.

Cao, L. and Tay, F. (2003) Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks* 14(6): 1506–1518.

Cortes, C. and Vapnik, V.N. (1995) Support vector networks. *Machine Learning* 20(1): 1–25.

Dunis, C., Laws, J. and Evans, B. (2008) Trading futures spread portfolios: Applications of higher order and recurrent networks. *European Journal of Finance* 14(5–6): 503–521.

Dunis, C., Laws, J. and Karathanasopoulos, A. (2011) Modeling and trading the Greek stock market with mixed neural network models. *Applied Financial Economics* 21(23): 1793–1808.

Dunis, C., Laws, J. and Sermpinis, G. (2009) The robustness of neural networks for modelling and trading the EUR/USD exchange rate at the ECB fixing. *Journal of Derivatives and Hedge Funds* 15(3): 186–205.

Giles, L.C. and Maxwell, T. (1987) Learning, invariance, and generalization in high-order neural networks. *Applied Optics* 26(23): 4972–4978.

Holland, J. (1995) *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA: MIT Press.

Howson, C. and Urbach, P. (1993) *Scientific Reasoning The Bayesian Approach*, 3rd edn. London: Open Course Publishing Company.

Huang, W., Nakamori, Y. and Wang, S. (2005) Forecasting stock market movement direction with support vector machine. *Computers & Operations Research* 32(10): 2513–2522.

Ince, H. and Trafalis, T. (2008) Short term forecasting with support vector machines and application to stock price prediction. *International Journal of General Systems* 37(6): 677–687.

Kaastra, I. and Boyd, M. (1996) Designing a neural network for forecasting financial and economic time series. *Neurocomputing* 10(10): 215–236.

Kerthi, S. and Lin, C.J. (2003) Asymptotic behaviors of support vector machines with Gaussian Kernel. *Neural Computation* 15(7): 1667–1689.

Kiani, K. and Kastens, T. (2008) Testing forecast accuracy of foreign exchange rates: Predictions from feed forward and various recurrent neural network architectures. *Computational Economics* 4(32): 383–406.

Kim, K. (2003) Financial time series forecasting using support vector machines. *Neurocomputing* 55(1–2): 307–319.

Knowles, A., Hussain, A., El Deredy, W., Lisboa, P.G. and Dunis, C.L. (2011) Higher order neural networks with Bayesian confidence measure for the prediction of the EUR/USD exchange rate. In: M. Zhang (ed.) *Artificial Higher Order Neural Networks for Economics and Business*. New York: IGI Global, pp. 48–59.

Matias, J.M. and Reboredo, J.C. (2012) Forecasting performance of non-linear models for intraday stock returns. *Journal of Forecasting* 31(2): 172–188.

Min, S., Lee, J. and Han, I. (2006) Hybrid genetic algorithms and support vector machines for bankruptcy prediction. *Expert Systems with Applications* 31(3): 652–660.

Nguyen, T., Gordon-Brown, L., Wheeler, P. and Peterson, J. (2009) *GA-SVM Based Framework for Time Series Forecasting. ICNC 1*, Fifth International Conference on Natural Computation, pp. 493–498.

Panda, C. and Narasimhan, V. (2007) Forecasting exchange rate better with artificial neural network. *Journal of Policy Modelling* 29(2): 227–236.

Scholkopf, B. *et al* (1999) Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks* 10(5): 1000–1017.

Scholkopf, B. and Smola, A.J. (2002) *Learning with Kernels: Support Vector Machines, Regularization and Beyond*. Cambridge, MA: MIT Press.

Sermpinis, G., Laws, J. and Dunis, C.L. (2013) Modelling and trading the realised volatility of the FTSE100 futures with higher order neural networks. *European Journal of Finance*, pp. 1 –15, doi:10.1080/1351847X.2011.606990.

Shapiro, A.F. (2000) A hitchhiker's guide to the techniques of adaptive nonlinear models. *Insurance, Mathematics and Economics* 26(2–3): 119–132.

Vapnik, V.N. (2000) *The Nature of Statistical Learning Theory*. New York: Springer.

Wu, C., Tzeng, G. and Lin, R. (2009) A novel hybrid genetic algorithm for kernel function and parameter optimization in support vector regression. *Expert Systems with Applications* 36(3): 4725–4735.

Zhang, M., Shuxiang, X. and Fulcher, J. (2002) Neuron-adaptive higher order neural-network models for automated financial data modeling. *IEEE Transactions on Neural Networks* 13(1): 188–204.

Appendix A
ARMA Model

The outputs of our ARMA models used in this article are presented in Tables A1–A4.

Table A1: ARMA model – FTSE 100 daily

Dependent variable: FD
Method: Least squares
Date: 08/30/12 Time: 14:30
Sample (adjusted): 42 014
Included observations: 2011 after adjustments
Convergence achieved after 19 iterations
MA Backcast: 1 3

Variable	*Coefficient*	*Std. error*	*t-Statistic*	*Prob.*
C	−8.38E−05	0.000115	−0.729606	0.4657
AR(1)	−1.042120	0.101424	−10.27490	0.0000
AR(2)	−0.907030	0.114283	−7.936710	0.0000
AR(3)	−0.458089	0.098937	−4.630121	0.0000
MA(1)	0.987772	0.108827	9.076536	0.0000
MA(2)	0.798468	0.124746	6.400773	0.0000
MA(3)	0.282868	0.107148	2.639974	0.0084
R-squared	0.039213	Mean dependent var		−8.35E-05
Adjusted *R*-squared	0.036336	SD dependent var		0.005823
SE of regression	0.005716	Akaike info criterion		−7.487562
Sum squared resid	0.065480	Schwarz criterion		−7.468047
Log likelihood	7535.744	Hannan-Quinn criter.		−7.480399
F-statistic	13.63151	Durbin-Watson stat		1.994924
Prob(*F*-statistic)	0.000000			—

Table A2: ARMA model – FTSE 100 weekly

Dependent variable: FW
Method: Least squares
Date: 08/30/12 Time: 19:38
Sample (adjusted): 6403
Included observations: 398 after adjustments
Convergence achieved after 98 iterations
MA Backcast: OFF (Roots of MA process too large)

Variable	*Coefficient*	*Std. error*	*t-Statistic*	*Prob.*
C	−0.000627	0.000470	−1.332308	0.1835
AR(1)	−1.445518	0.066034	−21.89048	0.0000
AR(2)	−0.596176	0.134232	−4.441395	0.0000
AR(4)	0.398242	0.157095	2.535032	0.0116
AR(5)	0.434227	0.094513	4.594346	0.0000
MA(1)	1.494024	0.064204	23.27005	0.0000
MA(2)	0.710581	0.115780	6.137347	0.0000
MA(4)	−0.609238	0.134850	−4.517876	0.0000
MA(5)	−0.643614	0.083099	−7.745158	0.0000
R-squared	0.173177	Mean dependent var		−0.000360
Adjusted *R*-squared	0.156173	S.D. dependent var		0.011646
SE of regression	0.010698	Akaike info criterion		−6.215188
Sum squared resid	0.044519	Schwarz criterion		−6.125042
Log likelihood	1245.822	Hannan-Quinn criter.		−6.179482
F-statistic	10.18447	Durbin-Watson stat		2.004490
Prob(*F*-statistic)	0.000000			—

Table A3: ARMA model – ASE 20 daily

Dependent variable: AD
Method: Least squares
Date: 08/30/12 Time: 12:54
Sample (adjusted): 31 968
Included observations: 1966 after adjustments
Convergence achieved after 31 iterations
MA Backcast: 1 2

Variable	*Coefficient*	*Std. error*	*t-Statistic*	*Prob.*
C	9.63E-05	0.000299	0.322137	0.7474
AR(1)	0.384887	0.041772	9.214062	0.0000
AR(2)	−0.925718	0.038915	−23.78831	0.0000
MA(1)	−0.378925	0.042934	−8.825740	0.0000
MA(2)	0.925368	0.039997	23.13567	0.0000
R-squared	0.008176	Mean dependent var		0.000103
Adjusted *R*-squared	0.006153	S.D. dependent var		0.013243
SE of regression	0.013202	Akaike info criterion		−5.814375
Sum squared resid	0.341782	Schwarz criterion		−5.800174
Log likelihood	5720.531	Hannan-Quinn criter.		−5.809156
F-statistic	4.041508	Durbin-Watson stat		1.899201
Prob(*F*-statistic)	0.002878			—

Table A4: ARMA model – ASE 20 weekly

Dependent variable: ASW
Method: Least squares
Date: 08/31/12 Time: 19:33
Sample (adjusted): 4403
Included observations: 400 after adjustments
Convergence achieved after 26 iterations
MA Backcast: 1 3

Variable	*Coefficient*	*Std. error*	*t-Statistic*	*Prob.*
C	−0.001088	0.002816	−0.386274	0.6995
AR(1)	1.362917	0.035219	38.69793	0.0000
AR(2)	−1.298977	0.037701	−34.45453	0.0000
AR(3)	0.513189	0.032644	13.61207	0.0000
MA(1)	−1.380993	0.042239	−32.69454	0.0000
MA(2)	1.364629	0.036856	37.02560	0.0000
MA(3)	−0.916274	0.040735	−22.49367	0.0000
R-squared	0.077077	Mean dependent var		−0.000816
Adjusted *R*-squared	0.062986	S.D. dependent var		0.016803
SE of regression	0.016265	Akaike info criterion		−5.382228
Sum squared resid	0.103971	Schwarz criterion		−5.312378
Log likelihood	1083.446	Hannan-Quinn criter.		−5.354567
F-statistic	5.470158	Durbin-Watson stat		1.971396
Prob(*F*-statistic)	0.000019			—

Appendix B PERFORMANCE MEASURES

The performance measures are calculated as shown in Table B1.

Table B1: Trading simulation performance measures

Performance measure	*Description*
Annualized Return	$R^A = 252 * \frac{1}{N} \sum_{t=1}^{N} R_t \quad (27)$
	with $R_{\{t\}}$ being the daily return
Cumulative Return	$R^C = \sum_{t=1}^{N} R_t \quad (28)$
Annualized Volatility	$\sigma^A = \sqrt{252} * \sqrt{\frac{1}{N-1} * \sum_{t=1}^{N} (R_t - \bar{R})^2} \quad (29)$
Information ratio	$IR = \frac{R^A}{\sigma^A} \quad (30)$
	Maximum negative value of $\sum (R_t)$ over the period
Maximum drawdown	$MD = \underset{i=1,\cdots,t;t=1,\cdots,N}{Min} \left(\sum_{j=i}^{t} R_j \right) \quad (31)$

Appendix C PROPOSED METHODOLOGY AND HONNS CHARACTERISTICS

In Table C1, we present the characteristics of our proposed methodology and the characteristics of the HONNs with the best trading performance on the test sub-period for the different architectures are shown in Table C2.

Table C1: Proposed methodology parameters

Population size:	30
Selection type:	Roulette Wheel Selection
Elitism:	Best member of every population is maintained in the next generation
Crossover probability:	0.9
Mutation probability:	0.1

Table C2: Characteristics for higher order neural network

Parameters	*HONNs*
Learning algorithm	*Gradient descent*
Learning rate	0.001
Momentum	0.003
Iteration steps	1500
Initialization of weights	N(0,1)
Input nodes	14
Hidden nodes (1layer)	0
Output node	1

Appendix D NAÏVE BAYESIAN CLASSIFIER'S CHARACTERISTICS

We present in Table D1 the characteristics of the Naïve Bayesian classifier approach that were used in this article.

Table D1: Characteristics Naïve Bayesian approach

Inputs distributions	Normal (Gaussian) distribution was used for every input
Prior probabilities for the two classes	Prior probabilities are estimated from the relative frequencies of the classes in training
Densities interval	The density can extend over the whole real line