

# **Modeling and Trading the Greek Stock Market with Artificial Intelligence Models**

**Andreas Karathanasopoulos**



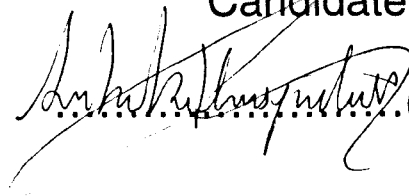
**Liverpool Business School**

A thesis submitted in partial fulfillment of the requirements  
of Liverpool John Moores University for the degree of  
Doctor of Philosophy

## ***Declaration***

I declare that with the exception of the assistance acknowledged, this dissertation is the result of an original investigation and that it has not been accepted or currently submitted in candidate for any other degree.

Candidate

.....(signed)

## ***Committee***

### ***Supervisory Team***

Prof. Christian Dunis (Director of Studies)

Jason Laws (1<sup>st</sup> Supervisor)

### ***Examination Team***

Prof. Spiridon Lukothanassis (External)  
(University of Patras, Department of Computer  
Engineering & Informatics)

Dr Chris Mulhearn (Internal)  
(Liverpool John Moores University, Liverpool Business  
School)

## Acknowledgments

I would like to take this opportunity to thank first and foremost, my Director of Studies; Professor Christian Dunis, whose help and guidance have been invaluable, over the past 2 years he has become a good friend.

I would also like to thank my Supervisor and long-time mentor Jason Laws for his help and guidance, especially in giving me the opportunity to start on this road.

Finally, I would like to thank my colleagues and friends Georgios Sermpinis and Kostandinos Theofilatos for their moral support over the past years.

**Abstract:**The main motivation for this thesis is to introduce some new methodologies for the prediction of the directional movement of financial assets with an application to the ASE20 Greek stock index. Specifically, we use some alternative computational methodologies named Evolutionary Support Vector Machine (ESVM), Gene Expression programming, Genetic Programming Algorithms and 2 hybrid combinations of linear and no linear models for modeling and trading the ASE20 Greek stock index using as inputs previous values of the ASE20 index and of four other financial indices. For comparison purposes, the trading performance of the ESVM stock predictor, Gene Expression Programming, Genetic Programming Algorithms and the 2 Hybrid combination methodologies have been benchmarked with four traditional strategies (a naïve strategy, a Buy and Hold strategy, a MACD and an ARMA models), and a Multilayer Perceptron (MLP) neural network model. As it turns out, the proposed methodologies produced a higher trading performance in terms of annualized return and information ratio, while providing information about the relationship between the ASE20 index and other foreign indices.

## **CHAPTER 1**

### **1.1 Introduction**

The development of accurate forecasting techniques is critical to economists, investors and risk analysts. This task is getting more complex as financial markets are getting increasingly interconnected and interdependent. The traditional statistical techniques, on which market forecasters were relying in previous years, seem to fail to capture this moving interrelationship among market variables. This context has led to a continuous search of techniques capable of identifying and capturing the nonlinearities, the discontinuities and the high frequency multi-polynomial

components characterizing financial time series today. Classes of such techniques that have provided promising results in previous years are Combinations of Neural Networks, Genetic Programming Algorithms, Gene Expression Programming and Support Vector Machines.

This thesis should be of interest to both hedgers and speculators who want to explore the use of alternative non linear models. An accurate prediction of the future stock market pattern will give them a considerable advantage and allow them to generate attractive return/risk profiles. Moreover, this thesis can contribute to the academic studies as it provides empirical evidents over the forecasting and trading abilities of a wide variety of non linear models over the mean of the ASE 20 Greek index. Also all the forecasts were evaluated through financial and trading criteria which makes it differ from most similar academic studies. Furhhermore, this thesis contributes to financial research by introducing a new technique that combines Support Vector machines with Genetic Algorithms providing the most profitable results.

## **1.2 Motivation of the Thesis**

The motivation of this thesis is to fill the hole in the literature and to provide empirical evidence of the utility of the models mentioned above in financial forecasting and trading applications. In order to achieve this, we benchmark our models not only with some traditional statistical and technical techniques but also with some other state-of-the-art NNs designs. Therefore, we will be able to validate if the theoretical advantages of our architectures compared to the more traditional NNs models are translated in more accurate/profitable forecasts. In order to achieve this our forecasts are evaluated through financial terms while in the literature most applications evaluate their financial forecasts only through statistical means. Moreover, we will

explore the utility of our architectures if we feed them not only with multivariate but also with autoregressive series as inputs. Furthermore we will be able to draw more solid conclusions on the forecasting ability of our models especially against our statistical autoregressive benchmarks as HONNs RNNs and MLP Neural Network models. Lastly, this research aims to provide the first empirical evidents over the forecasting power of Artificial intelligent models in a forecasting one day ahead context something that will further dinstiguish our research from previous similar studies and add originality to our application.

### **1.3 Contribution to the Knowledge**

In this dissertation we test and evaluate the forecasting and trading ability of the most promising new Neural Networks architectures combining them with autoregressive models (in our case ARMA model), Genetic Programming Algorithms, Gene Expression Programming and Support Vector Machines. We explore the utility of their performance in forecasting the mean in financial series. More specifically the contributions to knowledge of this dissertation are divided in 5 categories.

#### **1) Evaluating the forecasting and trading performance of Hybrid ARMA Neural Network.**

In chapter 4 we test and evaluate the performance of Hybrid Arma Neural Network in forecasting the ASE 20 Greek index using as inputs autoregressive series. In order to further improve the trading performance of our models we apply trading strategies using confirmation filters and leverage.

## **2) Evaluating the forecasting and trading performance of Mixed ARMA Neural Network.**

In chapter 5 we test and evaluate the performance of Mixed Arma Neural Network in forecasting the ASE 20 Greek index using as inputs autoregressive series. In order to further improve the trading performance of our models we apply trading strategies using confirmation filters and leverage.

## **3) Evaluating the forecasting and trading performance of Genetic Programming Algorithm.**

In chapter 6 we test and evaluate the performance of Genetic Programming Algorithm in forecasting the ASE 20 Greek index using as inputs autoregressive series. In order to further improve the trading performance of our models we apply trading strategies using confirmation filters and leverage.

## **4) Evaluating the forecasting and trading performance of Gene Expression Architecture.**

In chapter 7 we test and evaluate the performance of Gene Expression in forecasting the ASE 20 Greek index using as inputs autoregressive series. In order to further improve the trading performance of our models we apply trading strategies using confirmation filters and leverage.

## **5) Evaluating the forecasting and trading performance of Support Vector Machines.**

In chapter 8 we test and evaluate the performance of Support Vector Machines in forecasting the ASE 20 Greek index using as inputs autoregressive series. In order to further improve the trading performance of our models we apply trading strategies using confirmation filters and leverage.



## 1.4 Structure of the Dissertation

**Much of the content of this dissertation has either been accepted for publication, presented at conferences or has been submitted for publication at a peer-reviewed academic journals.**

Therefore the structure of the thesis comprises self contained chapters, each with its own focus. While the focus of each chapter follows a logical progression there may be some unavoidable repetitions between each chapter, however this has been kept in a minimum with each model only being described once. The references have been concentrated at the end of the thesis. The layout of the thesis is the presentation of 5 research papers. They make up the chapters as shown below:

**Chapter 4 – ‘Modelling and Trading the Greek Stock Market with Hybrid ARMA-Neural Network Models’.** This paper has been presented at the Forecasting Financial Markets 2009 conference in Luxembourg (23 to 26 May 2009) and after referees comments is currently in the last stage of the reviewing process for potential publication in *‘Quantitative Finance’*.

**Chapter 5 – ‘Modelling and Trading the Greek Stock Market with Mixed Neural Network Models’.** This paper has been presented at the Forecasting Financial Markets 2010 conference in Hannover (21 to 23 May 2010) and has been accepted for publication in the *‘Journal of Applied Financial Economics’*.

**Chapter 6 – ‘GP Algorithm versus Hybrid and Mixed Neural Networks’.** This paper has been presented at the Global Financial Markets 2010 conference in Azores (21

to 23 July 2010) and after referees comments is currently in the last stage of the reviewing process for potential publication in '*European Journal of Finance*'.

**Chapter 7** – 'Modelling and Trading the Greek Stock Market with Gene Expression and Genetic Programming Algorithms'. This paper will be presented at the second international conference of Finance in Rhodes (15 to 17 June 2011) and is currently being reviewed by '*European Journal of Forecasting*'.

**Chapter 8** – 'Stock Market Prediction Using Evolutionary Support Vector Machines: An Application to the ASE20 Index'. This paper has been presented at the Forecasting Financial Markets 2011 conference in Marseilles (27 to 29 May 2011) and is currently being reviewed by the '*European Journal of finance*' for potential publication.

## CHAPTER 2

### Literature Review

In this chapter we present the literature relevant to the Hybrid Neural Networks, Mixed Neural Networks, Genetic Programming Algorithm, Gene Expression Programming and Support Vector Machines models used on this thesis and the applications of them in financial forecasting context.

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 (Makridakis *et al.*(1982)) 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 (Makridakis (1989), Newbold *et al.* (1974), Palm *et al.* (1992)). Research in time series forecasting argues that predictive performance improves the combined models. (Bishop (1994), Clemen (1989), Hansen *et al.* (2003), Hibbert *et al.* (2000), Terui *et al.* (2002), Tseng *et al.* (2002), Zhang, (2003), Zhang *et al.* (2005)).

The reason for combining models comes from the assumption that either one cannot identify the true data generating process (Terui and Von Dyke. (2002)) or that a single model may not be sufficient to identify all the characteristics of the time series

(Zhang (2003)). Moreover the use of hybrid neural network has 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 the Hybrid GARCH-NN approach Wang (2007) and the Hybrid ARIMA/ARCH-NN of Fatima and Hussain (2008). Abraham *et al.* (2002) analysed the 24-month stock data for NASDAQ-100 main indices. Their hybrid system is Neuro-Fuzzy, a combination of neural network and fuzzy logic system. Lastly Andreou *et al.* (2006) propose knowledge-oriented neural network models combining nonparametric with parametric models (Black –Scholes) for option price data.

GP was first developed by Barricelli (1954) as evolutionary algorithms. Progressively into the 1960's and 1970's these 'evolutionary algorithms' became more commonly known and recognized as optimization methods. In particular, Rechenberg (1971) and his research team were able to solve complex engineering problems through the application of optimization methods as documented in his 1971 PhD thesis. Furthermore Holland (1975) was another influential figure in the 1970's However, Fogel *et al.* (1964) are among the earliest practitioners pioneering in GP methodology. They apply evolutionary algorithms to the problem of discovering finite-state automata. In the development of GP methodology it was later adapted to the Markov decision making process. More importantly the first evidence of GP as the 'tree based' method that we are familiar with in modern financial forecasting was provided by Cramer (1985). More recently, Cramer's work has been expanded further by John R. Koza (1990), Koza (1992), Koza (1994), Koza (1998) and Koza *et*

*al.* (1999, 2003) who apply these methodologies to complex optimization and search problems.

Although GP has now been established as a credible and respected technique this was not always the case. For example in the 1990's GP was considered incomprehensible. Enter the 2000's and the theory of GP has seen progressive and formidable growth. This has particularly been the case in the area of probabilistic models as GP has been incorporated with schema theories and Markov chain models. A variety of Genetic Programming applications is shown in the papers below: Winkler (2004), Winkler *et al.* (2004a, b), Madar *et al.* (2004, 2005), Willis *et al.* (1997), Tsang *et al.* (1998), Fukunaga and Stechert (1998) and Werner and Fogarty (2001).

GEP was first introduced by Ferreira (2001). Ferreira (2001, 2006) concludes saying that GEP is the latest addition to a family of Evolutionary Algorithms that provides financial practitioners with a further insight into artificial intelligence remedying the shortfalls attributed to traditional Genetic Algorithms (GAs) and Genetic Programming Algorithms (GP). In comparison to the GA and GP applications, the GEP proves to be superior due to the mere fact that it clearly distinguishes the differences between the genotype<sup>1</sup> and the phenotype<sup>2</sup> of individuals within a population. For instance, whilst a traditional GA classifies individuals as symbolic strings of fixed size (i.e. chromosomes) and GP classifies its individuals as non-linear comprising of different shapes and sizes (tree like structures); the GEP encompasses a combination of both. Hence Ferreira (2001) stresses that GEP represents not only an individual's genotype, in the form of chromosomes, but also

---

<sup>1</sup> The genotype is an individual's full hereditary information representing its exact genetic makeup.

<sup>2</sup> The phenotypes are the observed properties of an individual such as development and behaviour. These are largely determined from an individual's genotype.

its phenotype as a tree like structure of expressions in order to establish fitness. GEP is a new evolutionary technique and its applications so far is quite limited. However, it has been successfully applied in some real life problems like Lopez and Weinert (2004), Margny and El-Semman (2005) and Dehuri and Cho (2008).

The ESVM model combines genetic algorithms with Support Vector Machines. To the best of our knowledge, this is the first time that genetic algorithms have been combined with Support Vector Machines for the problem of modeling and trading financial indices. Until now many approaches have been based on Support Vector Machines for the modeling of financial time series. In 2003, two applications on SVM financial time series forecasting were developed. In Cao *et al.* (2003), SVMs are applied to the problem of forecasting several futures contracts from the Chicago Mercantile Market showing the superiority of SVMs over Back Propagation and regularized Radial Basis Function Neural Networks. In Kim (2003), SVMs are used to predict the direction of change in the daily Korean composite stock index and they are benchmarked against Back Propagation Neural Networks and Case Base Reasoning. The experimental results show that SVMs outperform the other methods and that they should be considered as a promising methodology for financial time-series forecasting. In 2005, Huang *et al.* use SVM for predicting the directional movement of the NIKKEI 225 index with very promising results. Lastly in Ince *et al.* (2008), Support Vector Regression is applied to the short-term forecasting of ten financial indices from the NASDAQ and outperforms all other traditional forecasting methods used.

The aim of the first 5 applications of this thesis is to provide empirical evidence around the utility of Combined Neural Networks, Genetic Programming Algorithms,

Gene Expression Programming and Evolutionary Support Vector Machines in forecasting the mean of financial series with autoregressive and multivariate series. In our research we evaluate our forecasts also with financial means and thus providing more solid conclusions around the financial utility of our models.

## **CHAPTER 3**

### **Models in the Study**

The forecasting methodologies used on this thesis are Neural Networks, Hybrid Neural Networks, Mixed Neural Networks, Genetic Programming Algorithms, Gene Expression Programming and Support Vector Machines. The Benchmark models used to compare our above architectures are ARMA model, MACD, a Naive strategy plus a Buy and Hold strategy. All these models are used to forecast the returns of the ASE 20 Greek index.

### **3.1 The Multi-Layer Perceptron**

A standard MLP 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 introduce the explanatory variables to the network, without any processing. 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 nonlinear activation function and passes it on to the output layer if the calculated value is above a threshold.

The network architecture of a 'standard' Multi-Layer Perceptron looks as presented in figure 1:

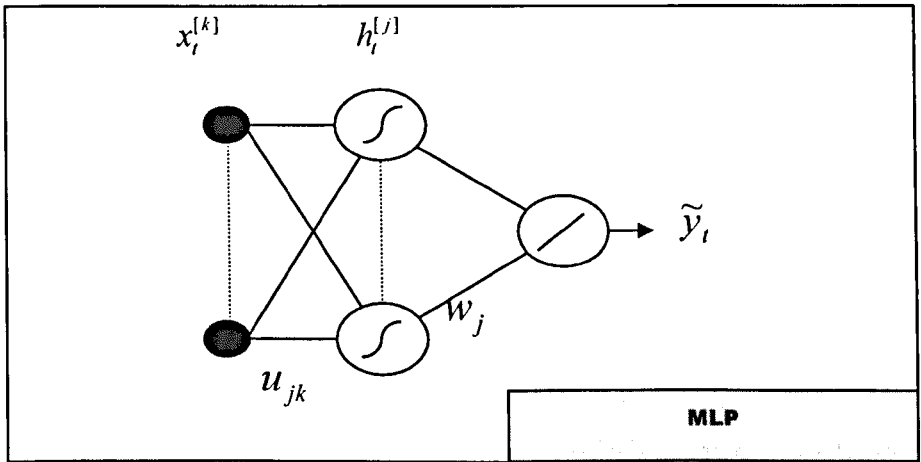


Fig. 1: A single output, fully connected MLP model

where:

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

$h_t^{[m]}$  ( $m = 1, 2, \dots, j + 1$ ) 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

$\textcircled{S}$  is the transfer sigmoid function:  $S(x) = \frac{1}{1 + e^{-x}}$ , [1]

$\textcircled{/}$  is a linear function:  $F(x) = \sum_i x_i$  [2]

The error function to be minimised is:

$$E(u_{jk}, w_j) = \frac{1}{T} \sum_{i=1}^T (y_i - \tilde{y}_i(u_{jk}, w_j))^2, \quad \text{with } y_i \text{ being the target value} \quad [3]$$

## 3.2 The Recurrent Network

Our next model is the recurrent neural network. While a complete explanation of RNN models is beyond the scope of this thesis, we present below a brief explanation of the significant differences between RNN and MLP architectures. For an exact specification of the recurrent network, see Elman (1990).

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 (1996) “the main disadvantage of RNNs is that they require substantially more connections, and more memory in simulation, than standard backpropagation networks” (p.569), 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.

### 3.2.1 The RNN Architecture

A simple illustration of the architecture of an Elman RNN is presented below.

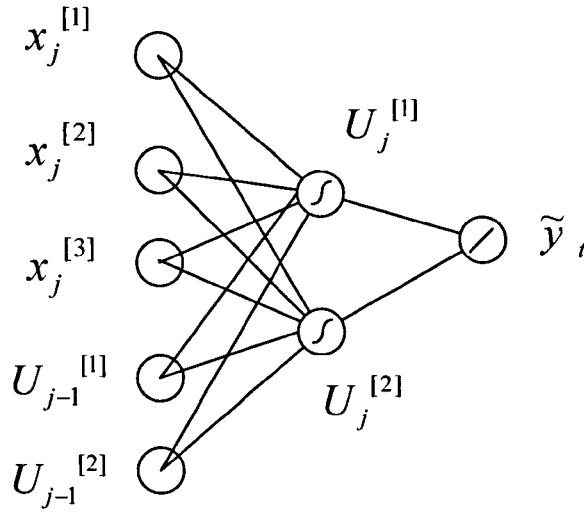


Fig. 2: Elman Recurrent neural network architecture with two nodes on the hidden layer.

where:

$x_t^{[n]}$  ( $n = 1, 2, \dots, k + 1$ ),  $u_t^{[1]}$ ,  $u_t^{[2]}$  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  $w_t^{[n]}$  ( $n = 1, 2, \dots, k + 1$ ) are the network weights

$U_t^{[f]}$  ( $f = 1, 2$ ) is the output of the hidden nodes at time  $t$

$\textcircled{S}$  is the transfer sigmoid function:  $S(x) = \frac{1}{1 + e^{-x}}$ , [4]

$\textcircled{/}$  is the linear output function:  $F(x) = \sum_i x_i$ , [5]

The error function to be minimised is:

$$E(d_t, w_t) = \frac{1}{T} \sum_{t=1}^T (y_t - \tilde{y}_t(d_t, w_t))^2 \quad [6]$$

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 the figure above).

### 3.3 Higher Order Neural Networks

Higher Order Neural Networks (HONNs) were first introduced by Giles and Maxwell (1987) and were called “Tensor Networks”. 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”. Moreover, HONNs are able to simulate higher frequency, higher order non-linear data, and consequently provide superior simulations compared to those produced by ANNs (Artificial Neural Networks)” (p. 188).

#### 3.3.1 The HONNs Architecture

While they have already experienced some success in the field of pattern recognition and associative recall<sup>3</sup>, HONNs have not yet been widely used in finance. The architecture of a three input second order HONN is shown below:

---

<sup>3</sup> Associative recall is the act of associating two seemingly unrelated entities, such as smell and colour. For more information see Karayiannis and Venetsanopoulos (1994).

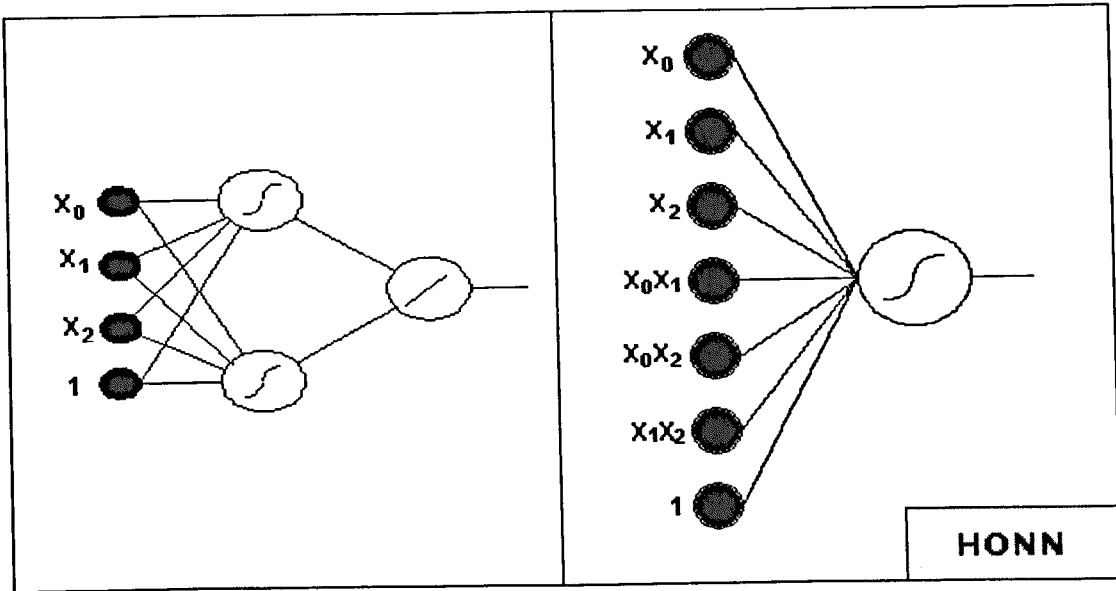


Fig. 3: Left, MLP with three inputs and two hidden nodes; right, second order HONN with three inputs

where:

$x_i^{[n]}$  ( $n = 1, 2, \dots, k + 1$ ) 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:  $S(x) = \frac{1}{1 + e^{-x}}$ , [7]



is a linear function:  $F(x) = \sum_i x_i$  [8]

The error function to be minimised is:

$$E(u_{jk}, w_j) = \frac{1}{T} \sum_{t=1}^T (y_t - \tilde{y}_t(u_{jk}))^2, \quad \text{with } y_t \text{ being the target value} \quad [9]$$

HONNs use joint activation functions; this technique reduces the need to establish the relationship between inputs when training. Furthermore this reduces the number

of free weights and means that HONNs can be faster to train than 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 can be largely avoided, Knowles *et al.* (2005). For a complete description of HONNs see Giles and Maxwell (1987).

### **3.4 Neural Networks and Hybrid Neural Networks**

Neural networks exist in several forms in the literature. The most popular architecture is the Multi-Layer 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 nonlinear activation function and passes it on to the output layer if the calculated value is above a threshold.

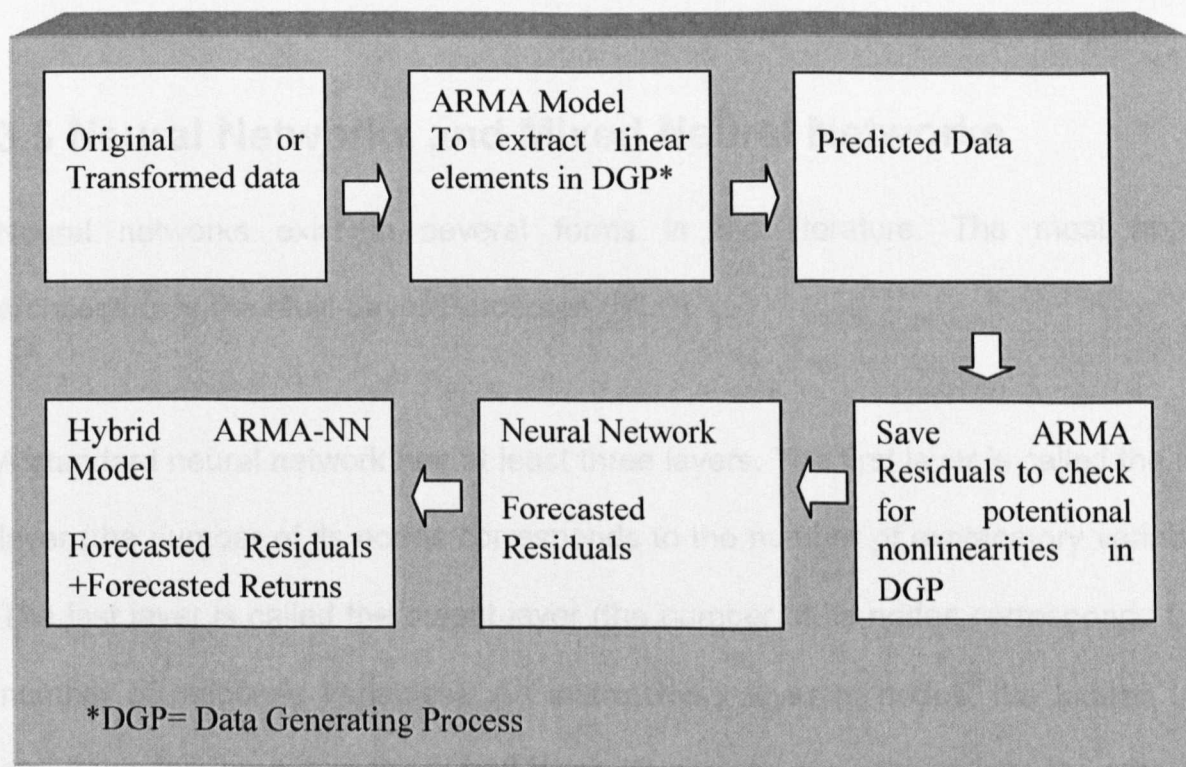
The training of the network (which is the adjustment of its weights in the way that the network maps the input value of the training data to the corresponding output value) starts with randomly chosen weights and proceeds by applying a learning algorithm called backpropagation of errors<sup>4</sup> (Shapiro (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). 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 be achieved by dividing the dataset into 3 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 dataset. Our networks, which are specially designed for financial purposes, will stop training when the profit of our forecasts in the test sub-period is maximized. Then the predictive value of the model is evaluated applying it to the validation dataset (out-of-sample dataset).

---

<sup>4</sup>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)).

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 (Hibon *et al.* (2005)).

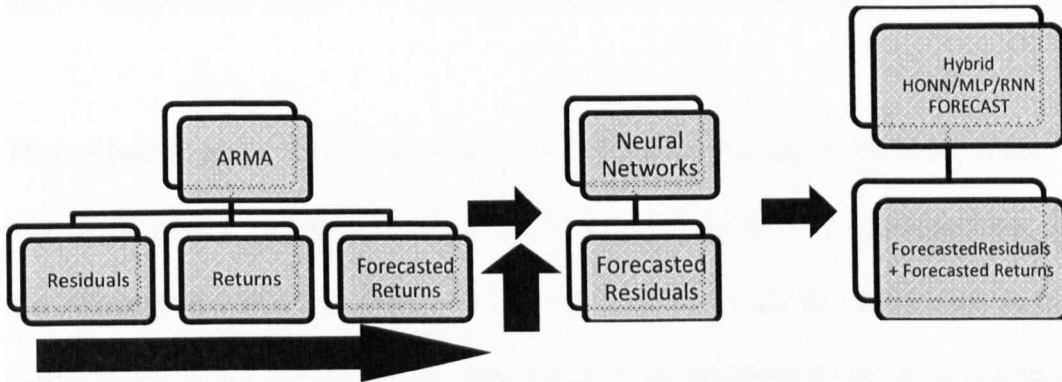
Combining methods involves using several redundant models designed for the same function, where the diversity of the components is thought important (Brown *et al.* 2005). The procedure of making a hybrid forecasting time series model can be achieved by combining an ARMA process in order to learn the linear component of the conditional mean pattern with an Artificial Neural Network process designed to learn its nonlinear elements. The construction of the hybrid ARMA-Neural Network model in details is in the figure below and in figure 5 (page 20)



**Fig. 4:** The architecture of Hybrid ARMA-Neural Network Model



### 3.4.1 THE HYBRID HONN, MLP, RNN, ARCHITECTURE



*Fig. 5:* The architecture of Hybrid ARMA-Neural Network Model

The methodology we follow to construct the Hybrid ARMA-Neural Network is divided in 3 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.

### 3.5 Neural Networks and Mixed Neural Networks

Neural networks exist in several forms in the literature. The most popular architecture is the Multi-Layer 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 nonlinear 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<sup>5</sup> (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). 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 be achieved by dividing the dataset into 3 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

---

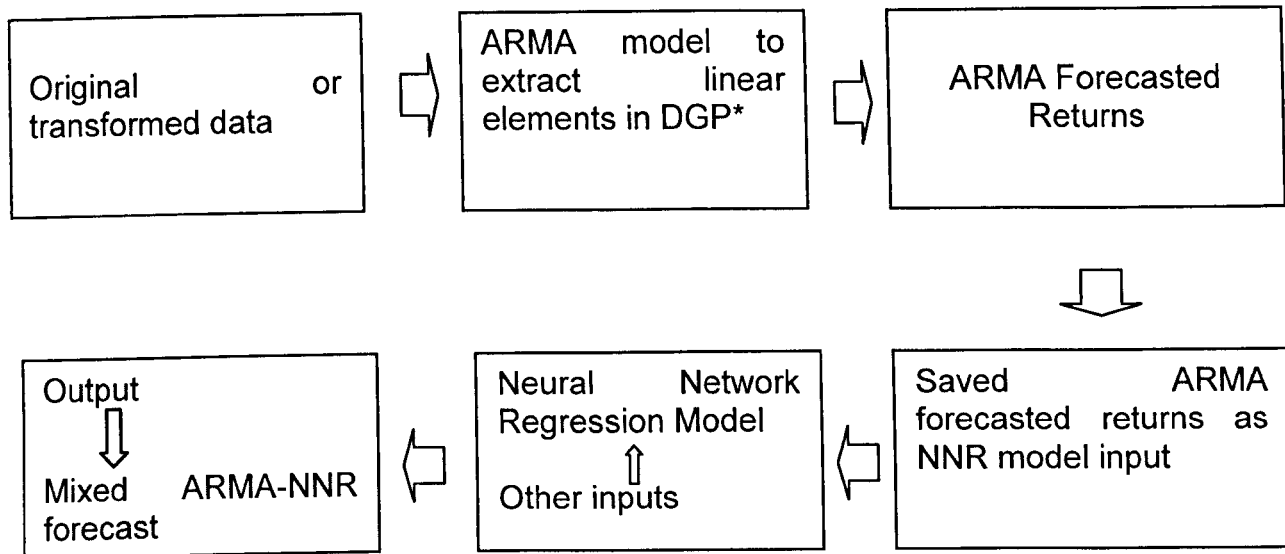
<sup>5</sup>Backpropagation networks are the most common multi-layer networks and are the most commonly used type in financial time series forecasting (Kaastra and Boyd (1996)).

(backpropagation of errors). The iteration length is optimised by maximising the forecasting accuracy for the test dataset. Our networks, which are specially designed for financial purposes, will stop training when the profit of our forecasts in the test sub-period is maximized. Then the predictive value of the model is evaluated applying it to the validation dataset (out-of-sample dataset).

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 (Hibon *et al.* (2005)).

Combining methods involves using several redundant models designed for the same function, where the diversity of the components is to be thought important (Brown *et al.* 2005). The procedure of making a mixed 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 nonlinear elements. The construction of the Mixed ARMA-Neural Network model is detailed in figure 6 below.

### **3.5.1 The Mixed HONN, MLP, RNN, Architecture**



\*DGP= Data Generating Process

*Fig. 6:* The architecture of Mixed Neural Network Model

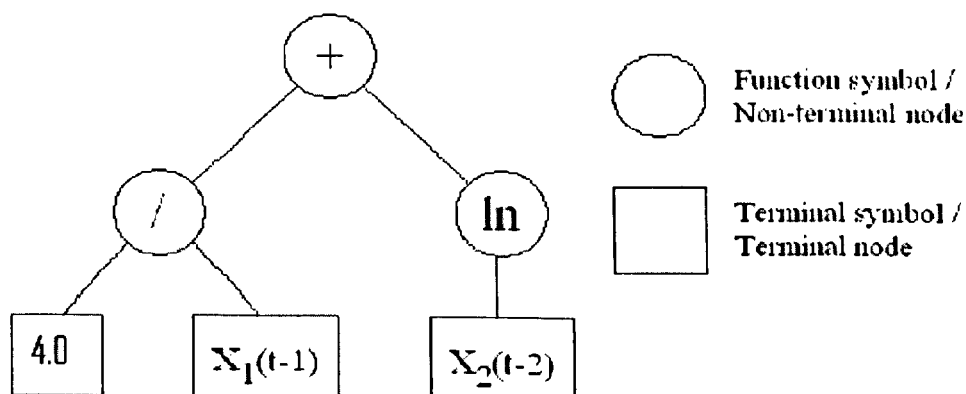
The methodology we follow to construct the Mixed ARMA-NNR model is divided into 2 steps. In the first step the ASE 20 index is modelled with a traditional ARMA model. In the second step the forecasted returns of the ARMA model are used as an input to the neural networks for forecasting the selected time series.

### 3.6 The Genetic Programming Algorithm

For the purpose of our research, the GP application is coded and implemented to evolve tree based structures that present models (sub trees) of input – output. In the design phase of our GP application we focused primarily on execution time optimization as well as limiting the ‘bloat effect’. The bloat effect is similar to the issue of overfitting experienced in Neural Networks however in our case we run a risk of continuously increasing and expanding the tree size. This algorithm is run in a steady state in that a single member of the population is replaced at a time. Furthermore, our GP application reproduces newer models replacing the weaker

ones in the population according to their fitness. Reasoning behind the decision to use a steady state algorithm is justified as they hold a greater selection strength and genetic drift over other algorithms such as a typical generational GAs. Additionally, steady state algorithms also offer exceptional multiprocessing capabilities.

In our application of the genetic programming we utilize formulas to evolve algebraic expressions that enable the analysis / optimization of results in a 'tree like structure'. This genetic tree structure consists of nodes (depicted as circles in the diagram below) which are essentially functions that perform actions within this structure. Furthermore, these functions are in place to generate output signals. On the other hand, the squares in the tree signify terminal functions representing the end of a function once the most superior sub tree (model) is achieved. For example, the below tree structure (model) is characterized by the algebraic expression  $4.0/x_1(t-1) + \ln(x_2(t-2))$ . In this case there is one output and the terminal nodes are constant at 4. Additionally, the outputs are expressed by  $x_1(t-1)$  and  $x_2(t-2)$ . In the execution of the genetic algorithm it has to be understood that each individual in the population correspond to a single sub tree structure. Each of these sub trees are limited by the predefined maximum tree size set to 6 in our application.



**Fig. 7:** Example of a tree structure

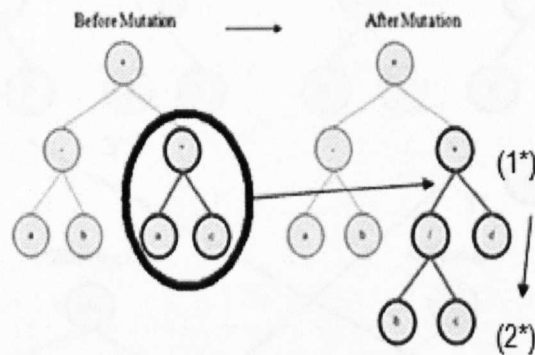
Koza (1998) summarises the functionality aspect of the GP algorithm in the following steps:

- (1) The generation of an initial population of randomly constructed models is developed with each model being represented in a tree like structure as discussed previously. Additionally, the evolutionary algorithm represents each chromosome of the population as a tree of variable length (i.e. total number of functions and terminals) or a maximum depth of the model tree. The process of randomly reproducing each variable of the population is completed once all of these functions of the tree are terminal symbols. However, until the process is halted by these 'terminal symbols' then the tree like structure of chromosomes continues to multiply (grow) with each generation as the population expands to not only include the parents but also their offspring. This is achieved by crossover and mutation operators. On the whole, it also has to be understood that the majority of these models produced in the initial population are, in most cases, unsatisfactory when tested for their performance with some individual models 'fitting' better than others. However, one of the virtues offered by Genetic Programming is that they exploit and manipulate these differences until the best fitting models, in terms of least error, are produced.
  
- (2) Following this initial generation of randomly selected models a random subset (sub tree) of the population is then selected for a tournament. Hence this process is known as a tournament selection phase. This process (tournament procedure) is essentially a selection mechanism to decipher which individuals from the population are to be selected for reproduction to develop the next generation.

- (3) An evaluation of the members of this subset is then carried out and assigned a fitness value. As stated by Koza (1998) the fitness cases are either selected at random or in some structured manner (e.g. at regular intervals). In our application, as mentioned briefly in the first step, the fitness value is defined as the mean squared error (MSE) with the lowest MSE being targeted as the best. Furthermore, the fitness may be measured in terms of the sum of the absolute value of the differences between the output produced by the model and/or the desired output (i.e. the Minkowski distance) or, alternatively, the square root of the sum of the squared errors (i.e. the Euclidean distance).
- (4) Following the establishment of fitness values the tournament winners are then determined. To reiterate, the winners of this scenario are the models with the lower MSE.
- (5) Having identified the tournament winners in the previous step we then proceed by exposing the models to two genetic operators known as mutations and crossovers. Both operators are discussed in more detail below:

*Mutation:* This is the creation of a new model that is mutated randomly from an existing one as circled in the diagram below (1\*). This one mutation point is indiscriminately chosen as an independent point and the resulting sub-tree is to be omitted. From this resulting sub-tree, another new sub-tree (2\*) is then reproduced using the same procedure that was initially implemented to create the original

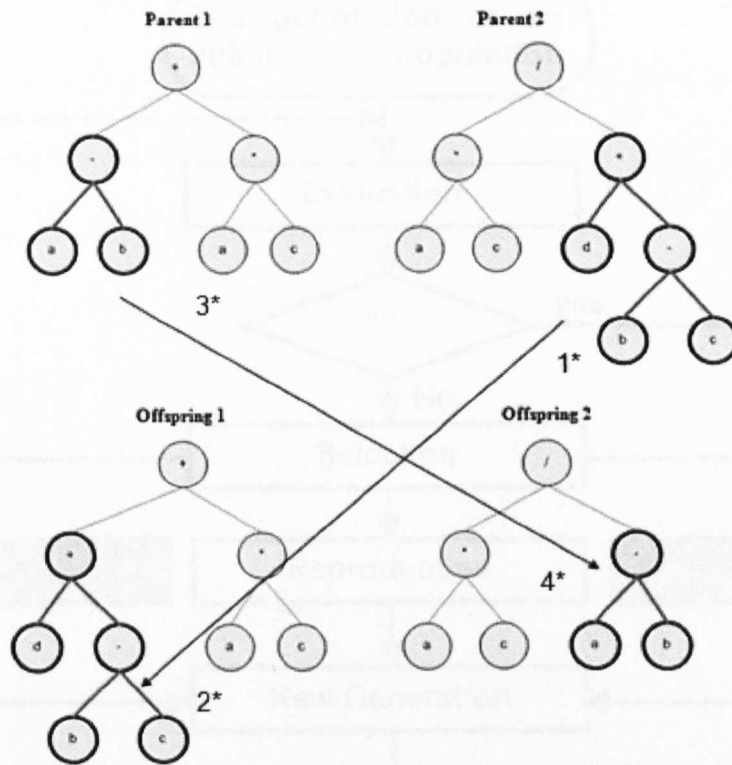
random population. Although this was the procedure we implemented for mutation there are also a number of alternative methods that are explored in other research.



*Fig. 8: Mutation tree structure example*

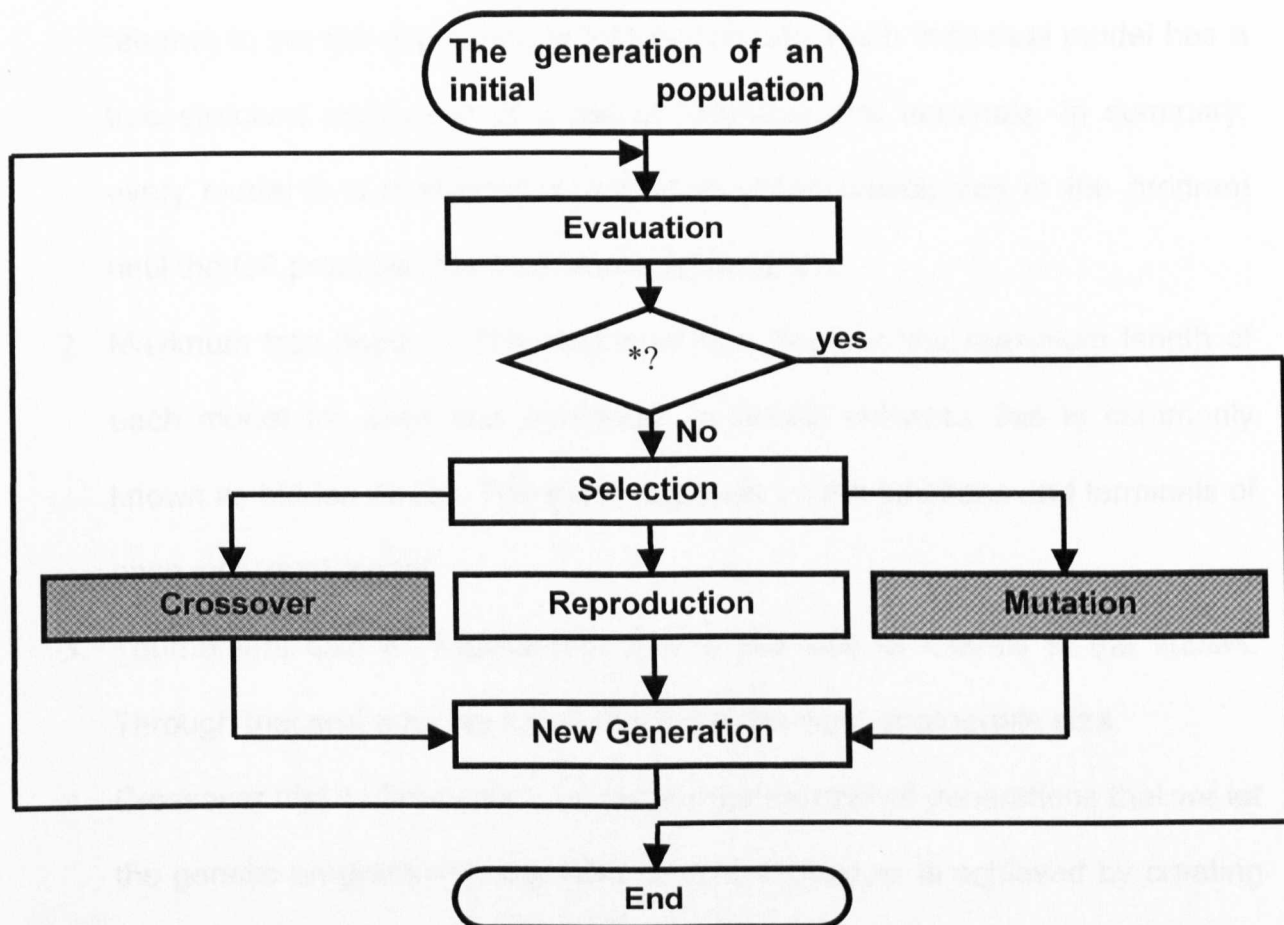
Crossover. This operator creates two new models from existing models by genetically recombining randomly chosen parts of them. This is achieved by using the crossover operation applied at a randomly chosen crossover point within each model. Due to the fact that entire sub-trees are swapped (from point 1\* to point 2\* and from points 3\* to 4\*), the crossover operation produces models as offsprings. Furthermore, the models are selected based on their fitness and the crossover allocates future trials to regions of the search space whose models contain parts from superior models. As a full explanation of crossovers is beyond the scope of this paper please refer to Koza (1992) for more details.





*Fig. 9: Crossover family tree like structure example*

- (6) The population is then altered with the tournament losers being replaced by the winners (superior) offspring.
  
- (7) Provided the termination criterion (depicted as the symbol '?' in the following flow of stages) is not reached, the algorithm returns to step 2 and these steps are repeated until the predefined termination criterion for genetic programming is satisfied. In our study we have set the termination criterion to 100,000 at which point the cycles are stopped and forecasted results can be obtained.
  
- (8) Ultimately, this protocol produces the best individual (model) of the population as a result.



\*note: the symbol '?' is the termination criterion which iterates or terminates the procedure of GP.

*Fig. 10:* The architecture of Genetic Programming Algorithm

### 3.6.1 Settings for Genetic Programming Parameters (See Appendix A.1.10)

The parameters used for the optimization of our individual models are defined in order to yield better results and are specified as follows:

1. Population size 200. The population size is the total number of randomly chosen models in our experiment. This number can be altered however in our specific case we found that it was more beneficial (in terms of annualised

returns) to set the population to 200 individuals. Each individual model has a tree structure composed of a set of functions and terminals. In summary, every model is a mathematical equation which participates in the program until the GP produces the best individual program.

2. Maximum tree depth 6. The maximum tree depth is the maximum length of each model (of each tree structure). In neural networks this is commonly known as hidden nodes. The depth depends on the functions and terminals of each individual model.
3. Tournament size 4. Tournament size is the size of models in the subset. Through trial and error we found this to be the most appropriate size.
4. Crossover trial 1. Crossover trial means the number of generations that we let the genetic programming algorithm to run. Crossover is achieved by creating two new offspring models for the new population by randomly recombining parts from two selected parents. In this experiment we have one crossover trial per generation.
5. Mutation probability 0.75. The mutation probability is the probability that can mutate parts of individual models from an existing one. Specifically mutation is performed by randomly selecting a parent with a probability related to its fitness, after that mutation randomly changes one or more genes representing part of the solution it encodes. Due to the fact that the population is 200 models, we use a relatively large probability. The mutation probability extends from an initial 0.1 and finishes at 0.9.

### 3.7 The Gene Expression Programming

As mentioned before the models in GEP are symbolic strings of fixed length representing an organism's genome (chromosome/genotype), but these simple entities are encoded as non linear entities of different sizes and shapes determining an organism's fitness (expression trees/phenotype). GEP chromosomes are made up of multiple genes spanning equal lengths across the structure of the chromosome. Each gene is comprised of a head (detailing symbols specific to functions and terminals) and a tail (only includes terminals). For a mathematical representation please refer to the below:

$$t = (n-1)h + 1 \quad [10]$$

Where:

$h$  = the head length of the gene.

$t$  = the tail length of the gene.

$n$  = total number of arguments within the function<sup>6</sup> (maximum arity)

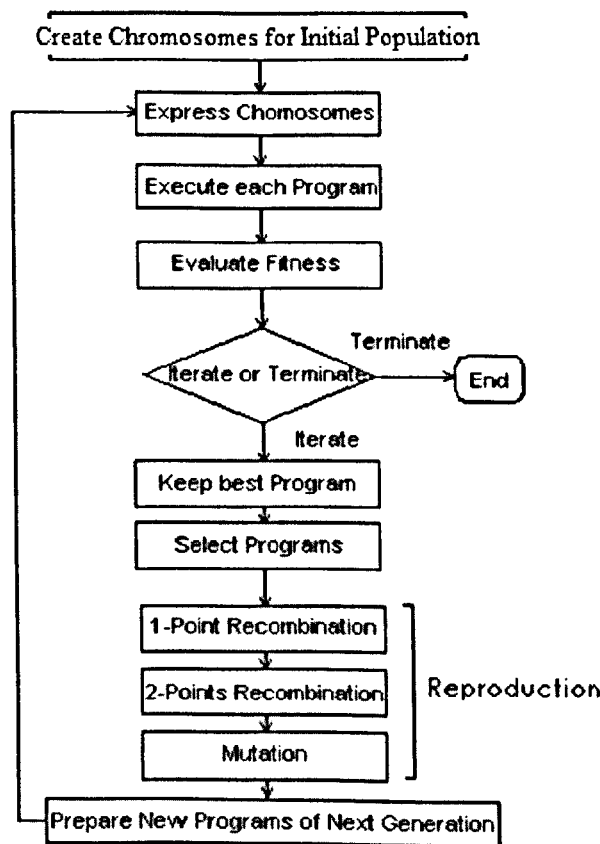
As an inference the set of terminals included within both the heads and tails of the chromosomes contain constants as well as case specific variables. In addition, regardless of the fact that each of the genes is equal and fixed in size they hold the capacity to code for multiple and varied expression trees (ET). For example, the structure of GEP is able to cope in circumstances when the first element of a gene is terminal producing a single node as well as when multiple nodes ('sub-trees' reproduced by functions) are produced in search for eventual terminality. In contrast

---

<sup>6</sup> This is determined by the user. In most cases a function will either be a Boolean function or any mathematical function that is suited to a specific problem.

with its predecessors, GEP does not require the rejecting of invalid individuals from the population, as valid ETs are always generated. Thus, each gene encodes an ET and in situations where multiple generation arise, GEP codes for sub ETs with interlinking functions to enable reproduction of generations. Furthermore, the expression of each ET is enabled by an Open Reading Frame (ORF) which assists in the decoding process. Additionally, while the ORF is initiated at the beginning of each gene it has to be understood that the eventual terminal points are not always determined to be located at the end of the gene

Although it is crucial to understand the workings of a GEP it is also just as important to understand its step by step process of evolution. This is depicted in the diagram below.



*Fig. 11: Flow chart of Gene Expression Algorithm*

The different steps of the algorithm from the above diagram are explained in more detail as follows:

### *1. Creation of Initial Population*

Similar to other evolutionary algorithms, GEP randomly generates an initial population from populations of individuals and all succeeding populations are spawned from this initial population. In the spawning of new generations genetic operators evolve each of the individuals by 'mating' them with other individuals in the population. These genetic operators are deciphered by the nature of the problem which one wants to solve. Genetic operators may include (but are not limited to) '+', '-', '\*' and '/' symbols for mathematical models and 'And', 'Or', 'Nand', 'Nor', 'Xor', 'Nxor', '<', '>', '< or =', and '> or =' for logical expressions as explained by Ferreira (2001). Therefore, the terminals and functions (symbols) may vary from problem to problem. Other intricacies such as gene size also have to be specified by the user at this stage.

### *2. Express chromosomes*

In this step we progress by developing expression trees from our chromosomes. The structure of each ET is in such a way that the root or the first node corresponds with beginning of each gene. The resulting offspring evolved from the first node is dependent on the number of arguments. In this process of evolution the functions may have numerous arguments however the terminals have an arity of zero. Each of the resulting offspring's characteristics is populated in nodes ordered from left to right. This process is concluded once terminal nodes are established.

### *3. Execute each program*

We are now ready to generate the initial population and develop resulting ETs. This is explained by Ferreira (2001).

### *4. Evaluate fitness*

In order to create an accurate model suited to our forecasting requirements it is imperative that a function which minimizes error and improves accuracy is used. Therefore, in order to evolve our initial population in line with our target market we must clearly define the goal of our model. Ultimately, this 'fitness function' determines the optimality of our solution. In our application, as mentioned before in the GP algorithm, the fitness value is defined as the mean squared error (MSE)] with the lowest MSE being targeted as the best. However, on the odd occasion some of the individuals that are generated randomly to create our initial population provide suitable solutions and hence arrive at terminal functions. More often than not though, individuals for the initial population provide poor candidates for the purpose of the investigation and require further evolution to achieve terminal values.

### *5. Keep best Program*

In our GEP model the main principal during the process of evolution is the generation of offspring from two superior individuals to achieve 'elitism'. As a consequence the best individuals from the parent generation produce offsprings in future generations with the most desirable traits whilst the individuals with less desirable traits are removed. On this basis our model minimizes error and maintains superior forecasting abilities. As explained in greater detail by Ferreira (2001), elitism is the cloning of the best chromosome(s)/individual(s) to the next

population (also called generation). Furthermore, the role of 'elitism' (via suited genetic operators) enables the selection of fitter individuals without eliminating the entire population.

## *6. Selection*

The selection of individuals based on their 'fitness' is carried out during the 'tournament' selection for reproduction and modification. This process selects the individuals at random with the superior ones being chosen for genetic modification in order to create new generations. The intensity of competition is dictated by the tournament size which is adjusted and set by the practitioner. The greater the tournament size the more competitive the selection process and therefore weaker individuals are less likely to compete.

## *7. Reproduction*

In the reproduction of future generations we have to consider the types of genetic operators which make this 'evolution' possible. Specifically we apply the genetic operators known as mutation and recombination as explained below.

*Mutation:* This is the creation of a new model that is mutated randomly from an existing one as circled in the first diagram below. This one mutation point is indiscriminately chosen as an independent point and the resulting chromosome is to be omitted. From this resulting chromosome, another new chromosome is then reproduced using the same procedure that was initially implemented to create the original random population. Although this was the procedure we implemented for



mutation there are also a number of alternative methods that are explored in other research like the second diagram below.

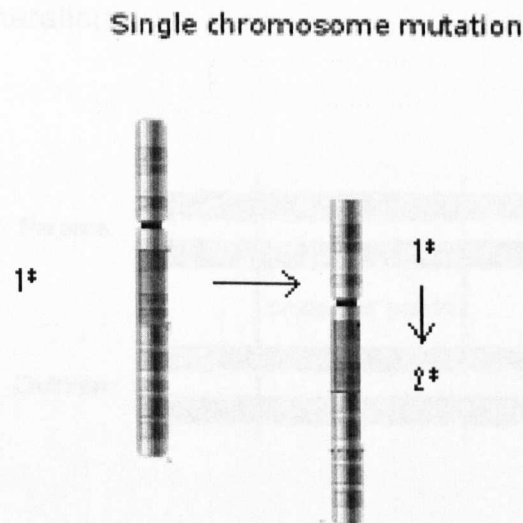


Fig. 12: Mutation chromosome structure example

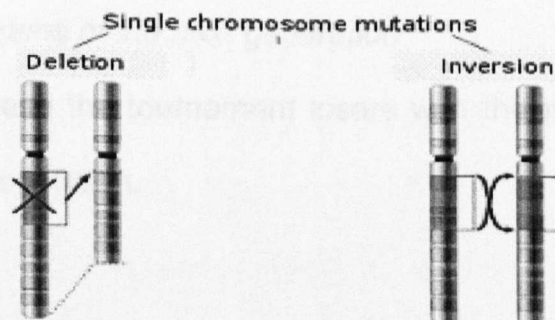


Fig. 13: Mutation chromosome structure examples

- **Recombination:** in contrast to our mutation operator this process is not executed at random. Instead the parent chromosomes are matched and split up or 'spliced' at identical points in order to determine recombination points. The subsequent spliced parts of each of the genes are then exchanged between the two selected chromosomes on the basis of probability. This

results into two new individuals as a result of genetic engineering. Note that during reproduction it is the chromosomes of the individuals, not the expression trees (ETs) that are reproduced with modification and transmitted to the next generation.

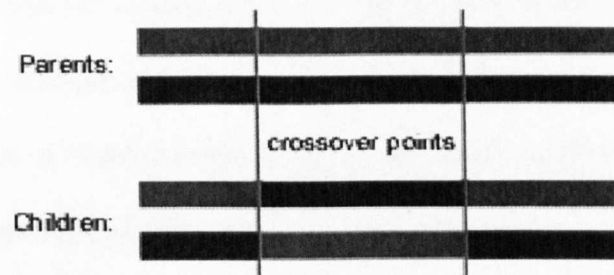


Fig. 14: Chromosome recombination structure example

#### 8. Prepare new programs of the next generation

At this step, we replace the tournament losers with the new individuals created by reproduction in the population.

#### 9. Termination criterion

We check if the termination criterion is fulfilled, if it is not we return to step 2. As a termination criterion we used a maximum number of 100.000 generations during which the GEP was left to run.

#### 10. Results

As a result we return the best individual ever found during the evolution process.

### **3.7.1 Settings for Gene Expression Programming Parameters (See Appendix A.1.10)**

1. Population size 1000. The population size is the total number of randomly chosen models in our experiment. This number can be altered however in our specific case we found that it was more beneficial (in terms of annualised returns) to set the population to 1000 individuals. Each individual model has a tree structure composed of a set of functions and terminals. In summary, every model is a mathematical equation which participates in the program until the GEP produces the best individual program.
2. Tournament size 20. Tournament size is the size of models in the subset. Through trial and error we found this to be the most appropriate size.
3. Type of recombination: two points recombination. In two-point recombination the parent chromosomes are paired and two points are randomly chosen by which both chromosomes are split. The material between the recombination points is then exchanged between the two chromosomes, forming two new daughter chromosomes.
4. Head length 50. The head length is the size of the part of the GEP chromosome which is called "head". The value of this parameter determines the maximum size of the models that the GEP will produce.
5. Mutation probability 0.75. The mutation probability is the probability that can mutate parts of individual models from an existing one. Specifically mutation is performed by randomly selecting a parent with a probability related to its fitness, after that mutation randomly changes one or more genes representing part of the solution it encodes. Due to the fact that the population is 1000

models, we use a relatively large probability. The mutation probability is tested over a range from 0.1 to 0.9.

### **3.8 Support Vector Machines**

Support vector machines (SVM) 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, SVM have already been applied in many prediction and classification problems in finance and economics (Ince *et al.* (2008), Cao *et al.* (2003), Huang *et al.* (2005), Kim (2003)) 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 SVM 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 section 3.8.1.

SVM 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 section 3.8.2.

### 3.8.1. Linear separability of data and Linear SVMs

Suppose we are given a set of examples  $(x_1, y_1), \dots, (x_l, y_l)$ , where  $x_i \in \mathbb{R}^N$  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 (which is not usually the case). In this case, we can find an optimal weight vector  $w_0$  such that  $\|w_0\|^2$  is minimum (in order to maximize the margin  $\Delta = 2/\|w_0\|$  of separation (Scholkopf *et al.* (1999)) and  $y_i * (w_0 * x_i + b_0) \geq 1, i = 1, \dots, l$ .

The support vectors are the training examples  $x_i$  that satisfy the equality, i.e.  $y_i(w_0 * x_i + b_0) = 1$ . 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  $a_i$  (Lagrange multipliers) in Vapnik (2000):

$$W(a) = \sum_{i=1}^l a_i - \frac{1}{2 \sum_{i=1}^l \sum_{j=1}^l a_i * a_j * y_i * y_j * \langle x_i, x_j \rangle} \quad [11]$$

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

$$F(x) = \text{sgn}\{w_0 * x + b_0\} \quad \text{where} \quad w_0 = \sum_{i=1}^l a_i * y_i * x_i \quad [12]$$

and the sum only takes into account  $N_s \leq l$  nonzero support vectors (i.e. training set vectors  $x_i$  whose  $a_i$  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  $x$ .

### 3.8.2 Non-linear Separability of data and Non-Linear SVMs

Cases in which the data is not linearly separable, like in financial modeling problems, are handled by introducing slack variables  $(\xi_1, \xi_2, \dots, \xi_l)$  with  $\xi_i \geq 0$  such that  $y_i \cdot (w \cdot x_i + b_0) \geq 1 - \xi_i, i = 1, \dots, l$ . The introduction of the variables  $\xi_i$  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:

$$\text{Minimize } \frac{1}{2} w * w + C * \sum_{i=1}^l \xi_i \quad [13]$$

subject to

$$y_i(w \cdot x_i + b_0) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0, i = 1, \dots, l \quad [14]$$

The control of the learning capacity is achieved by the minimization of the first part of Eq. (13) while 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 at the same time increasing the generalization ability of our classifier.

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  $\Phi$  chosen *a priori* (Cortes *et al.* (1995), Scholkopf *et al.* (1999)). The optimal separating hyperplanes are then constructed in this space.

The corresponding optimization problem is obtained from Eq. (11) by substituting  $\mathbf{x}$  by its mapping  $\mathbf{z} = \Phi(\mathbf{x})$  in the feature space, i.e. the maximization of  $W(\alpha)$ . Also, the constraint  $0 \leq \alpha_i$ , becomes  $0 \leq \alpha_i \leq C$  (assuming the nonseparable case). When it is possible to derive a proper kernel functional  $K$  such that  $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$  the mapping  $\Phi$  is not explicitly used. Conversely, given a symmetric positive kernel  $K(x, y)$ , Mercer's theorem, in Scholkopf *et al.* (2002), states that there exists a mapping  $\Phi$  such that  $K(x, y) = \langle \phi(x_i), \phi(x_j) \rangle$ . By designing a kernel  $K$  that satisfies Mercer's condition, the training algorithm is reformulated to the maximization of

$$W(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \cdot \alpha_j \cdot K(x_i, x_j) \cdot y_i \cdot y_j \quad [15]$$

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

$$F(\mathbf{x}) = \text{sgn} \left( \sum_{i=1}^l \alpha_i \cdot y_i \cdot K(\mathbf{x}, x_i) + b_0 \right) \quad [16]$$

With different expressions for inner products  $K(\mathbf{x}, x_i)$  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, i.e.,

$$K(\mathbf{x}, x_i) = (\langle \mathbf{x}, x_i \rangle + 1)^d \quad [17]$$

and the Radial Basis Function kernel has the form

$$K(\mathbf{x}, x_i) = \exp(-\text{gamma} \cdot \|\mathbf{x} - x_i\|^2) \quad [18]$$

The Radial basis function (RBF) kernels construct decision functions of the form:

$$F(\mathbf{x}) = \text{sgn} \left( \sum_{i=1}^l \alpha_i \cdot y_i \cdot \exp(-\text{gamma} \cdot \|\mathbf{x} - x_i\|^2 + b_0) \right) \quad [19]$$

In the case of the RBF kernel type, the SVM training algorithm determines the centers (support vectors)  $x_i$ , the corresponding weights  $a_i$  and the threshold  $b_0$ . This kernel nonlinearly maps samples into a higher dimensional space so 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 Radial Basis Kernel Function because of its higher reliability in finding optimal classification solutions in most practical situations (Kerthi *et al.* (2003)). The second reason is the number of hyperparameters which 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.

### **3.8.3 The Evolutionary SVM (ESVM) Stock Predictor**

In this section, we describe the proposed methodology. The ESVM stock predictor 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 for the first time which are a heuristic evolutionary technique known for its 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, in 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 which are optimized using GA are the parameters  $C$  and  $\gamma$  used by Support Vector Machines. As described in the previous section the parameter  $C$  is a kind of regularization parameter, that controls the tradeoff between learning capacity and training set errors and  $\gamma$  is a parameter of the RBF Kernel function.

For the genetic algorithm used in our wrapper methodology, the one-point crossover and the mutation operators were used. One-point crossover creates two offspring from every two parents. The parents are selected at random, a crossover point  $c_x$ , is selected at random, and two offspring 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.

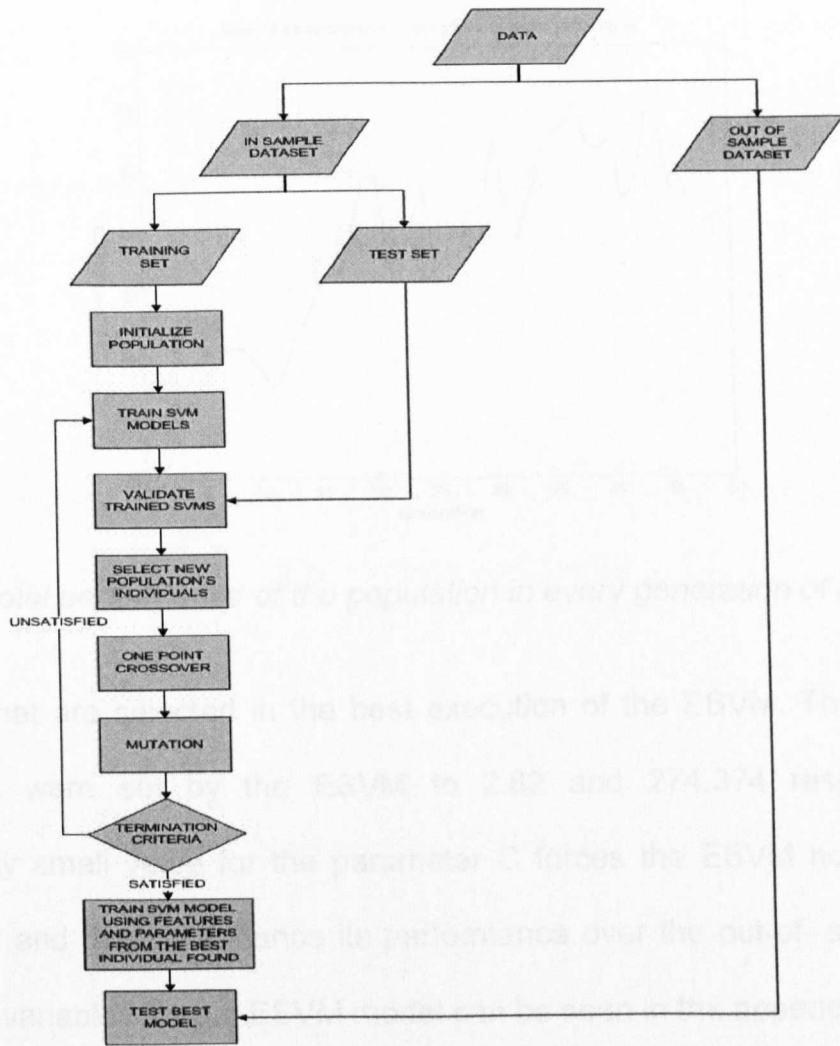
For the selection step of the GA, roulette selection (Holland (1995)) was used. In roulette selection chromosomes are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. In our approach, elitism was used to raise the evolutionary pressure in better solutions and to accelerate the evolution. By using elitism, we assured that the best solution is copied without changes to the new population, so the best solution found can survive at the end of every generation. The fitness function is defined as in equation [20]:

$$fitness = accuracy + accumulated\_return \quad [20]$$

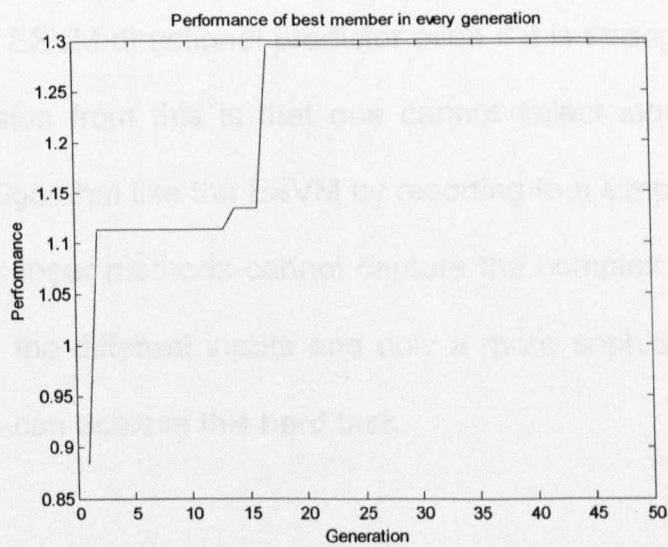
where *accuracy* is the SVM accuracy in the in sample test set and *accumulated\_return* is the accumulated return of the SVM in the sample test set. We chose this fitness function in order to balance the accuracy and financial effectiveness of the classifiers. The size of the initial population was 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% away from the best fitness of the current population. Specifically, when the average fitness across the current population is less than 5% 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 15.

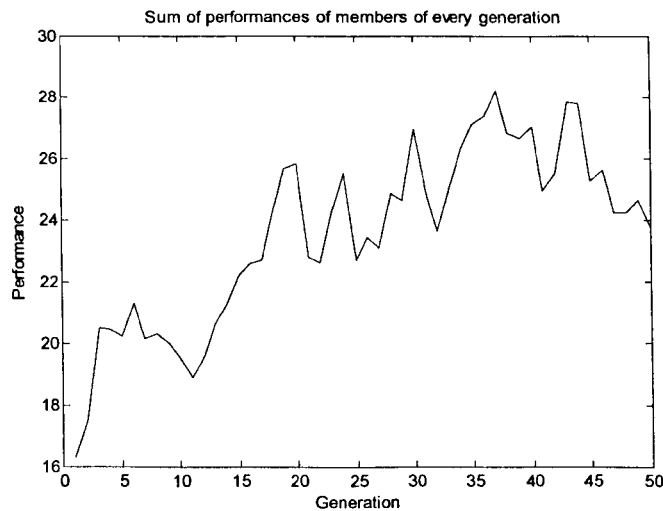
The ESVM is applied to the problem of forecasting the one day ahead direction of ASE-20 Greek stock index and then trading it. In Figure 15 we present an example of the performance of the best member of the population in every generation. In the vertical axes the performance is measured using the fitness function (20). In Figure 16 we present the total performance of the population in every generation of a single run. In order to compute the total performance of every population, the performances of the individuals (using equation 20) are summed.. From Figures 16 and 17 one can easily observe that our methodology converges after approximately 25 generations and that using different termination criteria would not improve our performance.



**Fig. 15:** Flowchart of ESVM methodology



**Fig. 16:** Performance of the best member of the population in every generation of a single run



*Fig. 17: Total performance of the population in every generation of a single run*

The inputs that are selected in the best execution of the ESVM. The parameter C and gamma were set by the ESVM to 2.82 and 274.374 respectively. This comparatively small value for the parameter C forces the ESVM not to overfit the training data and thus to enhance its performance over the out-of- sample dataset. Explanatory variables for our ESVM model can be seen in the appendix. (A.1.2)

By further examining our results we conclude that data from the FTSE100 index are not retained by the ESVM directional predictor even if it is strongly related to ASE20 index. The conclusion from this is that one cannot select inputs to be used in a machine learning algorithm like the ESVM by resorting to a simple method like linear correlation. Simple linear methods cannot capture the complex multiple correlations that exist between the different inputs and only a more sophisticated and powerful technique like GAs can achieve this hard task.

### 3.9 Benchmark Models

In this chapter, we benchmark our neural network models with 4 traditional strategies, namely an autoregressive moving average model (ARMA), a moving average convergence/divergence technical model (MACD) a naïve strategy and a buy and hold strategy.

### 3.9.1 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:

$$\hat{Y}_{t+1} = Y_t \quad [21]$$

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.

### 3.9.2 Moving Average

The moving average model is defined as:

$$M_t = \frac{(Y_t + Y_{t-1} + Y_{t-2} + \dots + Y_{t-n+1})}{n} \quad [22]$$

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<sup>7</sup>.

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.

### 3.9.3 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)<sup>8</sup>.

The ARMA model takes the form:

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \dots + \phi_p Y_{t-p} + \varepsilon_t - w_1 \varepsilon_{t-1} - w_2 \varepsilon_{t-2} - \dots - w_q \varepsilon_{t-q} \quad [23]$$

where  $Y_t$  is the dependent variable at time  $t$

$Y_{t-1}$ ,  $Y_{t-2}$ , and  $Y_{t-p}$  are the lagged dependent variable

$\phi_0$ ,  $\phi_1$ ,  $\phi_2$ , and  $\phi_p$  are regression coefficients

$\varepsilon_t$  is the residual term

$\varepsilon_{t-1}$ ,  $\varepsilon_{t-2}$ , and  $\varepsilon_{t-p}$  are previous values of the residual

<sup>7</sup>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.

<sup>8</sup> For a full discussion on the procedure, refer to Box et al. (1994) or Pindyck and Rubinfeld (1998).

$w_1$ ,  $w_2$ , and  $w_q$  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 Appendix A.1.1).

The selected ARMA model takes the form:

$$Y_t = 2.90 \cdot 10^{-4} + 0.376Y_{t-1} - 0.245Y_{t-3} - 0.679Y_{t-7} + 0.374\varepsilon_{t-1} - 0.270\varepsilon_{t-3} - 0.677\varepsilon_{t-7} \quad [24]$$

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<sup>9</sup>.

### 3.9.4 Buy and Hold strategy

Buying the index (asset) at the beginning of the review period and selling it back at the end.

$$\text{BHret.} = R_t$$

### 3.10.1. The ASE 20 Greek Index and Related Financial Data

For chapters 4 – 7 we use the same data period and for the sake of conciseness we have provided reference to appendix 1 for data segregation (A.1.3), a histogram of returns (A.1.5), a graph of our total dataset (a.1.4) and finally the explanatory variables used as inputs for each of our models (A.1.2). In chapter 8 we expand our dataset to include 2009 and 2010 data and as a result we directly refer to them in our text.

---

<sup>9</sup> Statistical measures are given in section 4.3 below.



For Futures contracts 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 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 thesis<sup>10</sup>. For further insight into our data segregation and a graph of our entire data set please refer to the appendix (A.1.3).

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. It is

---

<sup>10</sup> 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.

also non-stationary and we decided to transform the ASE 20 series into stationary series of rates of return<sup>11</sup>. A histogram depicting a summary of statistics can be seen in appendix A.1.5.

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

$$R_t = \left( \frac{P_t}{P_{t-1}} \right) - 1 \quad [25]$$

As inputs to our networks and based on the autocorrelation function and some ARMA experiments we selected 2 sets of autoregressive and moving average terms of the ASE 20 returns and the 1-day Riskmetrics volatility series. Explanatory variables for our traditional neural networks as well as our Hybrid Neural Networks are presented in the appendix (A.1.2)

In order to train the neural networks we further divided our dataset as seen in appendix A.1.3.

---

<sup>11</sup> 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.

## CHAPTER 4

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

### Overview

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 a 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.

## 4.1. Introduction

The use of intelligent systems for market predictions has been widely established. This chapter deals with the application of hybridized 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 paper investigates methods capable of identifying and capturing all the discontinuities, the nonlinearities and the high frequency multipolynomial components characterizing 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 thesis 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.* (2004) and Dunis *et al.* (2008b) where HONNs also demonstrate a forecasting superiority on the EUR/USD series over more traditional techniques such as a MACD and a naïve strategy. However, the RNN which performed remarkably well, show 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.

## 4.2 Methodology

A complete description of our Hybrid Neural Network architectures used on this application plus the benchmark models is on chapter 3. 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 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 Pindyck and Rubinfeld (1998), Theil (1966) and Dunis and Chen (2005) while their mathematical formulas are on Appendix A.1.6. 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 the table below we present our results for the out of sample period.

	NAIVE	MACD	ARMA	MLP	RNN	HONN
<b>RMSE</b>	0.0329	0.0254	0.0239	0.0470	0.0241	0.0240
<b>MAE</b>	0.0234	0.0174	0.0161	0.0163	0.0170	0.0299
<b>MAPE</b>	811.13%	393.44%	115.00%	106.97%	275.23%	679.96%
<b>THEIL-U</b>	0.6863	0.7534	0.9446	0.9661	0.8287	0.7289

	Hybrid-MLP	Hybrid-RNN	Hybrid-HONN
<b>RMSE</b>	0.0238	0.0237	0.0237
<b>MAE</b>	0.0160	0.0160	0.0159
<b>MAPE</b>	113.19%	112.83%	113.00%
<b>THEIL-U</b>	0.8891	0.8873	0.8868

*Table 1: Out-of-sample statistical performance*

As can be seen in Appendix A.1.7 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 although the difference with the other models is very small. 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.

### 4.3 Empirical Trading Simulation Results

The trading performance of all the models considered in the validation subset is presented in the table below. 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. Appendix A.1.8 provides the performance of all the NNs in the training and the test sub-periods while Appendix A.1.9 and A.1.6 provide the characteristics of our networks and the performance measures respectively. 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 2 below, the Hybrid-RNN has a lower performance compared to the Hybrid-MLP model and Hybrid-HONN.

	NAIVE	MACD	ARMA	MLP	RNN	HONN
<i>Information Ratio (excluding costs)</i>	0.32	0.46	0.20	0.60	0.59	0.70
<i>Annualised Volatility (excluding costs)</i>	36.70%	38.12%	38.13%	38.11%	38.11%	38.10%
<i>Annualised Return (excluding costs)</i>	11.42%	17.63%	7.68%	22.99%	22.51%	26.75%
<i>Maximum Drawdown (excluding costs)</i>	-49.41%	-50.63%	-36.50%	-36.26%	-36.22%	-38.71%
<i>Positions Taken (annualised)</i>	119	38	72	105	147	98

	Hybrid-MLP	Hybrid-RNN	Hybrid-HONN
<i>Information Ratio (excluding costs)</i>	0.86	0.81	0.94
<i>Annualised Volatility (excluding costs)</i>	38.08%	38.09%	38.07%
<i>Annualised Return (excluding costs)</i>	32.80%	30.72%	35.67%
<i>Maximum Drawdown (excluding costs)</i>	-59.05%	-59.05%	-59.05%
<i>Positions Taken (annualised)</i>	94	93	94

*Table 2: Trading performance results*

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 nonlinear elements definitely appears to add value in this application.

### **4.3.1 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. (2008a), 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.

According to the Athens Stock Exchange, transaction costs for financial institutions and fund managers dealing 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 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.



	NAIVE	MACD	ARMA	MLP	RNN	HONN
<b>Information Ratio (excluding costs)</b>	0.32	0.46	0.20	0.60	0.59	0.70
<b>Annualised Volatility (excluding costs)</b>	36.70%	38.12%	38.13%	38.11%	38.11%	38.10%
<b>Annualised Return (excluding costs)</b>	11.42%	17.63%	7.68%	22.99%	22.51%	26.75%
<b>Maximum Drawdown (excluding costs)</b>	-49.41%	-50.63%	-36.50%	-36.26%	-36.22%	-38.71%
<b>Positions Taken (annualised)</b>	119	38	72	105	147	98
<b>Transaction costs</b>	16.66%	5.32%	10.08%	14.7%	20.58%	13.72%
<b>Annualised Return (including costs)</b>	-5.24%	12.31%	-2.4%	8.29%	1.93%	13.03%

	Hybrid-MLP	Hybrid-RNN	Hybrid-HONN
<b>Information Ratio (excluding costs)</b>	0.86	0.81	0.94
<b>Annualised Volatility (excluding costs)</b>	38.08%	38.09%	38.07%
<b>Annualised Return (excluding costs)</b>	32.80%	30.72%	35.67%
<b>Maximum Drawdown (excluding costs)</b>	-59.05%	-59.05%	-59.05%
<b>Positions Taken (annualised)</b>	94	93	94
<b>Transaction costs</b>	13.6%	13.02%	13.6%
<b>Annualised Return (including costs)</b>	19.2%	17.7%	22.07%

Table 3: Out-of-sample results with transaction costs

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 annualized return. It's 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.

### 4.3.2 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%<sup>12</sup> on the test data set.

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. (that is 0.016% per trading day<sup>13</sup>). Our final results are presented in table 4 below.

	NAIVE	MACD	ARMA	MLP	RNN	HONN
<i>Information Ratio (excluding costs)</i>	0.32	0.70	0.20	0.60	0.59	0.70
<i>Annualised Volatility (excluding costs)</i>	36.70%	40.03%	38.13%	40.28%	40.21%	40.31%
<i>Annualised Return (excluding costs)</i>	11.42%	18.51%	7.68%	24.30%	23.75%	28.30%
<i>Maximum Drawdown (excluding costs)</i>	-49.41%	-53.16%	-36.50%	-38.32%	-38.21%	-40.96%
<i>Leverage Factor</i>	-	1.050	-	1.057	1.055	1.058
<i>Positions Taken (annualised)</i>	119	38	72	105	147	98
<i>Transaction and leverage costs</i>	16.66%	5.6%	10.08%	15.02%	20.88%	14.04%
<i>Annualised Return (including costs)</i>	-5.24%	12.9%	-2.4%	9.28%	2.87%	14.26%

<sup>12</sup> 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 8 below.

<sup>13</sup> 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.

		Hybrid-MLP	Hybrid-RNN	Hybrid-HONN
<i>Information Ratio</i>	<i>(excluding costs)</i>	0.86	0.81	0.94
<i>Annualised Volatility</i>	<i>(excluding costs)</i>	40.14%	40.30%	40.24%
<i>Annualised Return</i>	<i>(excluding costs)</i>	34.57%	32.50%	37.71%
<i>Maximum Drawdown</i>	<i>(excluding costs)</i>	-62.24%	-62.48%	-62.46%
<i>Leverage Factor</i>		1.054	1.058	1.057
<i>Positions Taken</i>	<i>(annualised)</i>	94	93	94
<i>Transaction and leverage costs</i>		13.9%	13.34%	13.9%
<i>Annualised Return</i>	<i>(including costs)</i>	20.67%	19.16%	23.21%

*Table 4: Trading performance - final results*

As can be seen from table 4, 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.

## 4.4 Concluding Remarks

In this chapter, we apply Multi-layer Perceptron, Recurrent, Higher Order, Hybrid-Multilayer Perceptron, Hybrid-Recurrent and Hybrid-Higher Order Neural Networks to a one-day-ahead forecasting and trading task of the ASE 20 fixing series with only autoregressive terms as inputs. We use a naïve, a MACD and an ARMA model as benchmarks. We develop 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 nonlinear elements appears quite promising.

## CHAPTER 5

# Modelling and Trading the Greek Stock Market with Mixed Neural Network Models

### Overview

In this chapter, a mixed methodology that combines both the ARMA and NN models is proposed to take advantage of the unique strength of ARMA and NN models in linear and nonlinear modelling. Experimental results with real data sets indicate that the combined model can be an effective way to improve forecasting accuracy achieved by either of the models used separately. 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 Mixed Higher Order Neural Network, a Mixed Recurrent Neural Network and a Mixed Multilayer Perceptron Neural Network with some traditional techniques, either statistical such as a 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 Mixed-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 Mixed-MLP network produces better results and outperforms all other neural network and traditional statistical models in terms of annualised return. On the other hand the Hybrid-HONNs shows a superiority after all sophisticated strategies have been used in terms of annualised return as Dunis *et al.* (2010a) mention in a recent paper.

## 5.1 Introduction

The use of intelligent systems for market predictions has been widely established. This paper deals with the application of mixed 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 nonlinearities and the high frequency multipolynomial components characterizing 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 Mixed-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 Mixed Higher order Neural Network, a Mixed Recurrent Neural Network and a Mixed 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 Mixed-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, Mixed MLPs outperform all models in terms of annualised return. Our conclusion corroborates those of Lindemann *et al.* (2004) and Dunis *et al.* (2008b) where HONNs also demonstrate a forecasting superiority on the EUR/USD series over more traditional techniques such as a MACD and a naïve strategy. However, the RNN which performed remarkably well, show 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.

## **5.2 Methodology**

A complete description of our Mixed Neural Network architectures used on this application plus the benchmark models is on chapter 3. 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 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 Pindyck and Rubinfeld (1998), Theil (1966) and Dunis and Chen (2005) while their mathematical formulae are in Appendix A.1.6. For all four error statistics retained (RMSE, MAE, MAPE and Theil-U) the lower the output, the better the forecasting accuracy of the model concerned. In the table below we present our results for the out-of-sample period.

	NAIVE	MACD	ARMA	MLP	RNN	HONN
<b>RMSE</b>	0.0329	0.0254	0.0239	0.0470	0.0241	0.0240
<b>MAE</b>	0.0234	0.0174	0.0161	0.0163	0.0170	0.0299
<b>MAPE</b>	811.13%	393.44%	115.00%	106.97%	275.23%	679.96%
<b>THEIL-U</b>	0.6863	0.7534	0.9446	0.9661	0.8287	0.7289

	Mixed MLP	Mixed RNN	Mixed HONN
<b>RMSE</b>	0.0240	0.0512	0.0240
<b>MAE</b>	0.0162	0.0189	0.0163
<b>MAPE</b>	107.06%	135.13%	103.56%
<b>THEIL-U</b>	0.9762	0.7318	0.9826

*Table 5: Out-of-sample statistical performance*

As can be seen in Appendix A.1.7 for the in-sample period, Mixed-HONNs seems to 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 gives them an considerable advantage compared to the other models. Mixed-MLPs come second and Mixed-RNNs 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.

### 5.3 Empirical Trading Simulation Results

The trading performance of all the models considered in the validation subset is presented in the table below. We select the ARMA model with the higher profit in the in-sample period and 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. Appendix A.1.8 provides the performance of all the NNs in the training and the test sub-periods while Appendix A.1.9 and A.1.10 provide the characteristics of our networks and A.1.6 the performance measures. The Mixed-RNNs are trained with gradient descent as for the Mixed-MLPs. However, the increase in the number of weights, as mentioned before, makes the training process extremely slow: to derive our results, we needed for the mixed-RNNs about ten times the time needed with the Mixed-MLPs. As shown in table 6 below, the Mixed-RNN has a lower performance compared to the Mixed-MLP model and Mixed-HONN

	NAIVE	MACD	ARMA	MLP	RNN	HONN
<i>Information Ratio (excluding costs)</i>	0.32	0.46	0.20	0.60	0.59	0.70
<i>Annualised Volatility (excluding costs)</i>	36.70%	38.12%	38.13%	38.11%	38.11%	38.10%
<i>Annualised Return (excluding costs)</i>	11.42%	17.63%	7.68%	22.99%	22.51%	26.75%
<i>Maximum Drawdown (excluding costs)</i>	-49.41%	-50.63%	-36.50%	-36.26%	-36.22%	-38.71%
<i>Positions Taken (annualised)</i>	119	38	72	105	147	98

Mixed MLP	Mixed RNN	Mixed HONN
-----------	-----------	------------

<b>Information Ratio (excluding costs)</b>	0.83	0.78	0.91
<b>Annualised Volatility (excluding costs)</b>	38.08%	38.09%	38.07%
<b>Annualised Return (excluding costs)</b>	31.79%	29.63%	34.75%
<b>Maximum Drawdown (excluding costs)</b>	-26.29%	-27.94%	-28.20%
<b>Positions Taken (annualised)</b>	41	57	65

*Table 6: Trading performance results*

We can see that Mixed-HONNs perform significantly better than the Mixed-MLPs and the Mixed-NNs and significantly better than the standard neural network architectures. Learning first the linear component of the data generating process before applying a neural network to learn its nonlinear elements definitely appears to add value in this application. Comparing the recent paper of Dunis et al. (2010a) we notice that Hybrid-NNR models outperform in terms of information ratio Mixed-NN models. However much higher drawdowns, possibly linked to the higher trading frequency of the Hybrid models compared with the mixed models presented here.

### **5.3.1 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. (2008a), we check 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 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 Mixed 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 Mixed, 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.

### 5.3.2 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 is 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.

	NAIVE	MACD	ARMA	MLP	RNN	HONN
<i>Information Ratio (excluding costs)</i>	0.32	0.46	0.20	0.60	0.59	0.70
<i>Annualised Volatility (excluding costs)</i>	36.70%	38.12%	38.13%	38.11%	38.11%	38.10%
<i>Annualised Return (excluding costs)</i>	11.42%	17.63%	7.68%	22.99%	22.51%	26.75%
<i>Maximum Drawdown (excluding costs)</i>	-49.41%	-50.63%	-36.50%	-36.26%	-36.22%	-38.71%
<i>Positions Taken (annualised)</i>	119	38	72	105	147	98
<i>Transaction costs</i>	16.66%	5.32%	10.08%	14.7%	20.58%	13.72%
<i>Annualised Return (including costs)</i>	-5.24%	12.31%	-2.4%	8.29%	1.93%	13.03%

	Mixed MLP	Mixed RNN	Mixed HONN
<i>Information Ratio (excluding costs)</i>	0.83	0.78	0.91
<i>Annualised Volatility (excluding costs)</i>	38.08%	38.09%	38.07%
<i>Annualised Return (excluding costs)</i>	31.79%	29.63%	34.75%
<i>Maximum Drawdown (excluding costs)</i>	-26.29%	-27.94%	-28.20%
<i>Positions Taken (annualised)</i>	41	57	65
<i>Transaction costs</i>	5.74%	7.98%	9.1%
<i>Annualised Return (including costs)</i>	26.05%	21.65%	25.65%

*Table 7: Out-of-sample results with transaction costs*

We can see that, after transaction costs, the Mixed-MLP network outperforms all the other strategies based on the annualised return closely followed by the Mixed-HONN strategy. On the other hand, the naïve strategy and the ARMA model produce negative results after transaction costs are taken into account. The HONN and MACD achieve decent returns, yet well below those produced by our mixed ARMA-NN models.

### 5.3.3 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%<sup>14</sup> on the test data set.

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. (that is 0.016% per trading day<sup>15</sup>). Our final results are presented in table 8 below.

	NAIVE	MACD	ARMA	MLP	RNN	HONN
<b>Information Ratio (excluding costs)</b>	0.32	0.70	0.20	0.60	0.59	0.70
<b>Annualised Volatility (excluding costs)</b>	36.70%	40.03%	38.13%	40.28%	40.21%	40.31%

<sup>14</sup> 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 8 below.

<sup>15</sup> 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.

<b>Annualised Return (excluding costs)</b>	11.42%	18.51%	7.68%	24.30%	23.75%	28.30%
<b>Maximum Drawdown (excluding costs)</b>	-49.41%	-53.16%	-36.50%	-38.32%	-38.21%	-40.96%
<b>Leverage Factor</b>	-	1.050	-	1.057	1.055	1.058
<b>Positions Taken (annualised)</b>	119	38	72	105	147	98
<b>Transaction and leverage costs</b>	16.66%	5.60%	10.08%	15.02%	20.88%	14.04%
<b>Annualised Return (including costs)</b>	-5.24%	12.90%	-2.40%	9.28%	2.87%	14.26%

		Mixed-MLP	Mixed-RNN	Mixed-HONN
<b>Information Ratio (excluding costs)</b>		0.83	0.78	0.91
<b>Annualised Volatility (excluding costs)</b>		40.22%	40.22%	40.17%
<b>Annualised Return (excluding costs)</b>		33.57%	31.29%	36.67%
<b>Maximum Drawdown (excluding costs)</b>		-27.76%	-29.50%	-29.75%
<b>Leverage Factor</b>		1.056	1.056	1.055
<b>Positions Taken (annualised)</b>		41	57	65
<b>Transaction and leverage costs</b>		6.052%	8.30%	9.40%
<b>Annualised Return (including costs)</b>		27.51%	23.00%	27.27%

*Table 8: Trading performance - final results*

As can be seen from table 8, Mixed-MLPs continue to demonstrate a superior trading performance despite significant drawdowns. The Mixed-HONN strategy also performs well and presents the second highest annualised return. In general, we observe that all models are able to gain extra profits from the leverage as the increased costs are outweighed by the benefits of trading somewhat higher volumes. Again it is worth mentioning, that the time needed to train the HONN and the Mixed-HONN network was considerably shorter compared with that needed for the MLP, Mixed-MLP, RNN and the Mixed-RNN networks.

## 5.4 CONCLUDING REMARKS

In this chapter, we apply Multi-layer Perceptron, Recurrent, Higher Order, Mixed-Multilayer Perceptron, Mixed-Recurrent and Mixed-Higher Order neural networks to a one-day-ahead forecasting and trading task of the ASE 20 fixing series with only autoregressive terms as inputs. We use a naïve strategy, a MACD and an ARMA model as benchmarks. We develop 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 Mixed-HONNs demonstrates a higher trading performance in terms of annualised return and information ratio before transaction costs and more elaborate trading strategies are applied. When refined trading strategies are applied and transaction costs are considered the Mixed-MLPs manage to outperform all other models achieving the highest annualised return. The Mixed-HONNs and the Mixed-RNNs models perform remarkably as 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 Mixed-HONN network which presents a very close second best performance needs less training time than Mixed-RNN and Mixed-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 nonlinear elements appears quite promising.

## CHAPTER 6

### GP Algorithm versus Hybrid and Mixed Neural Networks

#### Overview

In the current chapter we present an integrated genetic programming environment, called java GP Modelling. The java GP Modelling environment is an implementation of the steady-state genetic programming algorithm. That algorithm evolves tree based structures that represent models of input – output relation of a system. The motivation of this chapter is to compare the GP algorithm with 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 the GP algorithm and 6 different ARMA-Neural Network combination designs representing a Hybrid, Mixed Higher Order Neural Network (HONN), a Hybrid, Mixed Recurrent Network (RNN), a Hybrid, Mixed classic Multilayer Perceptron (MLP) with some traditional techniques, either statistical such as a 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 time series closing prices over the period 2001-2008 using the last one and a half years for out-of-sample testing. We use the ASE 20 daily series 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 GP model does remarkably well and outperforms all other models in a simple trading simulation exercise. This is also the case when more sophisticated trading strategies using confirmation filters and leverage are applied, as the GP model still produces better results and outperforms all other neural network and traditional statistical models in terms of annualised return.

## 6.1 Introduction

The use of artificial intelligence for the purpose of forecasting market movements has been widely reviewed in academia. This study is a comparative analysis of the results yielded from utilizing a Genetic Programming Algorithm and various traditional Neural Network computing techniques when forecasting the Greek stock market. Additionally, we endeavour to develop more accurate and sophisticated techniques in order to increase the performance of our trading simulation. Due to the convergence and unification of global financial markets in recent years, this forecasting task has become increasingly challenging. Furthermore, traditional econometric methods on which forecasters have previously been reliant no longer satisfy the demands of market participants as they struggle to capture integrating features associated with today's markets. As discussed by Lisboa *et al* (2000), neural networks are an emergent technology with an increasing number of real-world applications offering a unique aspect to the world of financial forecasting. Nevertheless, some practitioners have tainted the virtues of neural networks with scepticism criticising their capacity to forecast and highlighting their limitations. Hence, this work investigates a new, contemporary and more proficient method of forecasting that is capable of identifying and dealing with discontinuities, nonlinearities and high frequency multi-polynomial components which are all



prevalent in financial series of today's markets. This model is most commonly known as the Genetic Programming (GP) algorithm.

GP Algorithms are domain-independent problem-solving techniques that are run in various environments. These environments are structured in a manner which approximates problems in order to produce forecasts at a high level of accuracy. GP can be categorized in the forecasting bracket known in the finance world as 'Evolutionary Algorithms'. The basis for this type of problem – solving technique derives from the Darwinian principle of reproduction and *survival of the fittest*. Additionally, GP is also similar to the biological genetic operations such as crossover and mutation. More importantly, Koza (1990, 1992) stress that GP addresses and quantifies complex issues as an automated process via programming, which enables computers to process and solve problems.

The Darwinian aspect of GP applies the theory of evolution to a population of computer programs of varying sizes and shapes. For instance, GP starts with an initial population of thousands or even millions of randomly generated computer programs. These programs comprise of programmatic elements built to apply the fundamental principles of biological evolution in order to create a new (and often improved) population of programs. As mentioned previously, the creation of this new population is generated in a domain-independent system applying the Darwinian theory of natural selection under the principal known as *survival of the fittest*. An analogue of the naturally-occurring genetic operation of sexual recombination (crossover), and occasional mutation, the crossover operation is designed to create syntactically valid offspring programs (given closure amongst the set of programmatic ingredients). GP combines the expressive high-level symbolic

representations of computer programs with the near-optimal efficiency of learning of Holland's (1975) genetic algorithm in order to produce highly accurate outputs. Koza (1998) mentioned that a computer program that solves or at the very least approximates a given problem often emerges from this process. Dissimilar to other models such as neural networks, GP does not require any prior knowledge of a model's structure for the purpose of system modelling. Alternatively, GP evolves a system model with parameter values that best fit specific data without manipulating the data to fit 'predefined' model structures as many other preceding forecasting methods tend to do. In other words, GP creates an initial population of models and evolves using genetic operators in order to calculate the mathematical expression which best fits the specified data input into the system. Furthermore, GP simultaneously searches for and refines a model's parameters and ultimately its structure.

The motivation for this chapter is to investigate the use of GP algorithm and several neural networks techniques combined with ARMA models in order to improve the forecasting performance using autoregressive terms as inputs. This is achieved by comparing six benchmark neural network combined architectures with a forecast produced by the GP Algorithm. Most notably, classic neural networks as Multilayer Perceptron (MLP), Higher Order Neural Network (HONN), Recurrent Neural Network (RNN), autoregressive moving average model (ARMA), or technical models such as a moving average convergence/divergence model (MACD), plus a naïve trading strategy are all reviewed as benchmark methods.

From the analysis it emerges that the GP algorithm demonstrates a remarkable performance and outperforms all other models in a simple trading simulation

exercise. This is also true when more sophisticated trading strategies are utilized with the application of confirmation filters and leverages as GP still demonstrates superior forecasting ability in terms of annualised return. It is worth mentioning the second best performance of the Hybrid HONNs and the Mixed HONNs. Dunis *et al.* (2010a, b) stress that the combination of neural networks can produce better forecasts compared with alternative techniques. Furthermore the Hybrid MLP and the Mixed MLP also perform well. Also, the RNNs which historically have performed remarkably well display less impressive forecasting potential in this research. It is observed that this might be due to the fact that they have an inability to provide accurate results when only autoregressive terms are used as inputs.

The remainder of the chapter is organised as follows. An overview of the different neural network models and Genetic Programming algorithm is given in section 6.2. Section 6.3 displays the empirical results of all the models considered and investigates the possibility of improving their performance with the application of more sophisticated trading strategies. Ultimately, Section 6.4 provides some concluding remarks.

## **6.2 Methodology**

A complete description of our Genetic Programming Algorithm used on this application plus the benchmark models is in chapter 3.

## **6.3 Empirical Trading Simulation Results**

The trading performance of all the models considered in the validation subset is presented in the table below. 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. Appendix A.1.8 provides the performance of all the NNs and the GP Algorithm in the training and the test sub-periods while both Appendix A.1.9 and A.1.10 provide the characteristics of our models and A.1.6 shows the performance measures. The Hybrid-RNNs, Mixed-RNNs are trained with gradient descent as for the Hybrid-MLPs and Mixed-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 and Mixed-MLPs. As shown in table 9 below, the Mixed-RNN, Hybrid-RNN have a slightly lower performance compared to the Hybrid-MLP, Hybrid-HONN, Mixed-MLP, Mixed-HONN and GP algorithm.

	NAIVE	MACD	ARMA	MLP	RNN	HONN
<i>Information Ratio (excluding costs)</i>	0.32	0.46	0.20	0.60	0.59	0.70
<i>Annualised Volatility (excluding costs)</i>	36.70%	38.12%	38.13%	38.11%	38.11%	38.10%
<i>Annualised Return (excluding costs)</i>	11.42%	17.63%	7.68%	22.99%	22.51%	26.75%
<i>Maximum Drawdown (excluding costs)</i>	-49.41%	-50.63%	-36.50%	-36.26%	-36.22%	-38.71%
<i>Positions Taken (annualised)</i>	119	38	72	105	147	98

	Hybrid MLP	Hybrid RNN	HybridHONN
<i>Information Ratio (excluding costs)</i>	0.86	0.81	0.94
<i>Annualised Volatility (excluding costs)</i>	38.08%	38.09%	38.07%
<i>Annualised Return (excluding costs)</i>	32.80%	30.72%	35.67%
<i>Maximum Drawdown (excluding costs)</i>	-59.05%	-59.05%	-59.05%
<i>Positions Taken (annualised)</i>	94	93	94

	Mixed MLP	Mixed RNN	Mixed HONN
<i>Information Ratio (excluding costs)</i>	0.83	0.78	0.91
<i>Annualised Volatility (excluding costs)</i>	38.08%	38.09%	38.07%
<i>Annualised Return (excluding costs)</i>	31.79%	29.63%	34.75%
<i>Maximum Drawdown (excluding costs)</i>	-26.29%	-27.94%	-28.20%
<i>Positions Taken (annualised)</i>	41	57	65

GP Algorithm	
<i>Information Ratio (excluding costs)</i>	1.03
<i>Annualised Volatility (excluding costs)</i>	38.04%
<i>Annualised Return (excluding costs)</i>	39.33%
<i>Maximum Drawdown (excluding costs)</i>	-28.20%
<i>Positions Taken (annualised)</i>	67

*Table 9: Trading performance results*

We can see that the GP algorithm performs significantly better than the Hybrid-HONNs, Hybrid-MLPs, Mixed HONNs, Mixed MLPs Hybrid-RNNs, and the Mixed-RNNs with similar sorts of drawdowns, and significantly better than the standard neural network architectures.

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.* (2008a), we check 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 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 Mixed 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 Mixed, MLP HONN models. This is not the case however but, in order to conserve space, these results are not shown here but they are available from the authors.

### 6.3.1 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 is 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 basis point=1/100 of 1%) or 0.14% per position.

	NAIVE	MACD	ARMA	MLP	RNN	HONN
<i>Information Ratio (excluding costs)</i>	0.32	0.46	0.20	0.60	0.59	0.70
<i>Annualised Volatility (excluding costs)</i>	36.70%	38.12%	38.13%	38.11%	38.11%	38.10%
<i>Annualised Return (excluding costs)</i>	11.42%	17.63%	7.68%	22.99%	22.51%	26.75%
<i>Maximum Drawdown (excluding costs)</i>	-49.41%	-50.63%	-36.50%	-36.26%	-36.22%	-38.71%
<i>Positions Taken (annualised)</i>	119	38	72	105	147	98
<i>Transaction costs</i>	15.47%	4.94%	9.36%	13.65%	19.11%	12.74%
<i>Annualised Return (including costs)</i>	-4.05%	12.69%	-1.68%	9.35%	3.40%	14.01%

	Hybrid MLP	Hybrid RNN	Hybrid HONN
<i>Information Ratio (excluding costs)</i>	0.86	0.81	0.94
<i>Annualised Volatility (excluding costs)</i>	38.08%	38.09%	38.07%
<i>Annualised Return (excluding costs)</i>	32.80%	30.72%	35.67%
<i>Maximum Drawdown (excluding costs)</i>	-59.05%	-59.05%	-59.05%
<i>Positions Taken (annualised)</i>	94	93	94
<i>Transaction costs</i>	12.22%	12.09%	12.22%
<i>Annualised Return (including costs)</i>	20.58%	18.63%	23.45%

	Mixed MLP	Mixed RNN	Mixed HONN
<i>Information Ratio (excluding costs)</i>	0.83	0.78	0.91
<i>Annualised Volatility (excluding costs)</i>	38.08%	38.09%	38.07%
<i>Annualised Return (excluding costs)</i>	31.79%	29.63%	34.75%
<i>Maximum Drawdown (excluding costs)</i>	-26.29%	-27.94%	-28.20%
<i>Positions Taken (annualised)</i>	41	57	65
<i>Transaction costs</i>	5.74%	7.98%	9.10%
<i>Annualised Return (including costs)</i>	26.05%	21.65%	25.65%

GP Algorithm	
<i>Information Ratio (excluding costs)</i>	1.03
<i>Annualised Volatility (excluding costs)</i>	38.04%
<i>Annualised Return (excluding costs)</i>	39.33%
<i>Maximum Drawdown (excluding costs)</i>	-28.20%
<i>Positions Taken (annualised)</i>	67
<i>Transaction costs</i>	9.40%
<i>Annualised Return (including costs)</i>	29.93%

*Table 10: Out-of-sample results with transaction costs*

We can see that, after transaction costs, the GP algorithm model outperforms all the other strategies based on the annualized net return closely followed by the Mixed-MLP, the Mixed HONN and the Hybrid HONNs strategy. On the other hand, the naïve strategy and the ARMA model produce negative results after transaction costs are taken into account. The HONN and MACD achieve decent returns, yet well below those produced by our best models.

### 6.3.2 Leverage to Exploit High Sharpe 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%<sup>16</sup> on the test data set.

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. (that is 0.016% per trading day<sup>17</sup>). Our final results are presented in table 11 below.

<sup>16</sup> 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 8.

<sup>17</sup> 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

	NAIVE	MACD	ARMA	MLP	RNN	HONN
<b>Information Ratio</b> <i>(excluding costs)</i>	0.32	0.70	0.20	0.60	0.59	0.70
<b>Annualised Volatility</b> <i>(excluding costs)</i>	36.70%	40.03%	38.13%	40.28%	40.21%	40.31%
<b>Annualised Return</b> <i>(excluding costs)</i>	11.42%	18.51%	7.68%	24.30%	23.75%	28.30%
<b>Maximum Drawdown</b> <i>(excluding costs)</i>	-49.41%	-53.16%	-36.50%	-38.32%	-38.21%	-40.96%
<b>Leverage Factor</b>	-	1.050	-	1.057	1.055	1.058
<b>Positions Taken</b> <i>(annualised)</i>	119	38	72	105	147	98
<b>Transaction and leverage costs</b>	15.47%	4.94%	9.36%	13.65%	19.11%	12.74%
<b>Annualised Return</b> <i>(including costs)</i>	-4.05%	13.57%	-1.68%	10.65%	4.64%	15.56%

	Hybrid-MLP	Hybrid-RNN	Hybrid-HONN
<b>Information Ratio</b> <i>(excluding costs)</i>	0.86	0.81	0.94
<b>Annualised Volatility</b> <i>(excluding costs)</i>	40.14%	40.30%	40.24%
<b>Annualised Return</b> <i>(excluding costs)</i>	34.57%	32.50%	37.71%
<b>Maximum Drawdown</b> <i>(excluding costs)</i>	-62.24%	-62.48%	-62.46%
<b>Leverage Factor</b>	1.054	1.058	1.057
<b>Positions Taken</b> <i>(annualised)</i>	94	93	94
<b>Transaction and leverage costs</b>	12.22%	12.1%	12.22%
<b>Annualised Return</b> <i>(including costs)</i>	12.35%	20.4%	24.89

	Mixed-MLP	Mixed-RNN	Mixed-HONN
<b>Information Ratio</b> <i>(excluding costs)</i>	0.83	0.78	0.91
<b>Annualised Volatility</b> <i>(excluding costs)</i>	40.22%	40.22%	40.17%
<b>Annualised Return</b> <i>(excluding costs)</i>	33.57%	31.29%	36.67%
<b>Maximum Drawdown</b> <i>(excluding costs)</i>	-27.76%	-29.50%	-29.75%
<b>Leverage Factor</b>	1.056	1.056	1.055
<b>Positions Taken</b> <i>(annualised)</i>	41	57	65
<b>Transaction and leverage costs</b>	6.052%	8.30%	9.40%
<b>Annualised Return</b> <i>(including costs)</i>	27.51%	23.00%	27.27%

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.



<b>GP Algorithm</b>	
<i>Information Ratio (excluding costs)</i>	1.03
<i>Annualised Volatility (excluding costs)</i>	41.84%
<i>Annualised Return (excluding costs)</i>	43.26%
<i>Maximum Drawdown (excluding costs)</i>	-31.02%
<i>Leverage Factor</i>	1.10
<i>Positions Taken (annualised)</i>	67
<i>Transaction and leverage costs</i>	9.95%
<i>Annualised Return (including costs)</i>	33.34%

Table 11: Trading performance - final results

As can be seen from table 11, the GP algorithm continues to demonstrate a superior trading performance despite significant drawdowns. The Mixed HONN, the Mixed MLP and the Hybrid HONN strategies also perform well and presents high annualised returns. In general, we observe that all models are able to gain extra profits from the leverage as the increased transaction costs countered by increased performance. Again it is worth mentioning, that the time needed to train the HONN, the Hybrid-HONN and the Mixed-HONN network was considerably shorter compared with that needed for the MLP, Hybrid-MLP, Mixed-MLP, RNN, Mixed-RNN and the Hybrid-RNN networks.

## 6.4 CONCLUDING REMARKS

In this chapter, we apply a Genetic Programming algorithm, Multi-layer Perceptron, Recurrent, Higher Order, Mixed-Multilayer Perceptron, Mixed-Recurrent, Mixed-Higher Order neural networks, Hybrid-Multilayer Perceptron, Hybrid-Recurrent, Hybrid-Higher Order neural networks to a one-day-ahead forecasting and trading task of the ASE 20 fixing series with only autoregressive terms as inputs. We use a

naïve strategy, a MACD and an ARMA model as benchmarks. We develop 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 GP algorithm demonstrates a higher trading performance in terms of annualised return and information ratio before transaction costs. When more elaborate trading strategies are applied and transaction costs are considered the GP algorithm again continues to outperform all other models achieving the highest annualised return. The Mixed-HONNs, the Mixed-RNNs and the Hybrid-HONNs models perform remarkably as well and seem to have ability in providing good forecasts when autoregressive series are only used as inputs.

It is also important to note that the Mixed-MLP network which presents a very close second best performance needs less training time than the GP algorithm, 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 strategies consisting of modelling in a first stage the linear component of a financial time series and then applying a neural network to learn its nonlinear elements and the use of Genetic Programming appear quite promising.

## CHAPTER 7

# Modelling and Trading the Greek Stock Market with Gene Expression and Genetic Programming Algorithms

### Overview

In this chapter we present an application of Gene Expression Programming Environment in modelling the ASE 20 Greek Index compared with an integrated genetic programming environment, called GP Modelling. The Gene Expression Programming (GEP) is a new evolutionary algorithm that evolves computer programs (they can take many forms: mathematical expressions, neural networks, decision trees, logical expressions and so on). The computer programs of GEP irrespective of their complexity are all encoded in linear chromosomes. Then the linear chromosomes are expressed or translated into expression trees. Thus, in GEP the genotype (the linear chromosomes) and the phenotype (the expression trees) are different entities (both structurally and functionally). The GP Modelling environment is an implementation of the steady-state genetic programming algorithm. That algorithm evolves tree based structures that represent models of input – output relation of a system. The motivation of this chapter is to compare the GP Algorithm with another new forecasting application named GEP Algorithm when applied to the task of predicting and trading the ASE 20 Greek Index using only autoregressive terms as inputs. This is done by benchmarking the forecasting performance of the GP and GEP with some more 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 and a Multilayer Perceptron neural Network.. More specifically, the

trading performance of all models is investigated in a forecast and trading simulation on ASE 20 time series closing prices over the period 2001-2008 using the last one and a half year for out-of-sample testing. We use the ASE 20 daily series 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 GEP model does remarkably well and outperforms all other models in a simple trading simulation exercise. This is also the case when more sophisticated trading strategies using confirmation filters and leverage are applied, as the GEP model still produces better results and outperforms the GP and traditional statistical models in terms of annualised return.

## **7.1 Introduction**

The use of artificial intelligence for the purpose of forecasting market movements has been widely reviewed in academia. This study is a comparative analysis of the results yielded utilizing by a Gene Expression Programming (GEP) Algorithm and a Genetic Programming (GP) Algorithm when forecasting the Greek stock market. Additionally, we endeavour to develop more accurate and sophisticated techniques in order to increase the performance of our trading simulation. However, due to the convergence and unification of global financial markets in recent years, this task has become increasingly challenging. Furthermore, traditional econometric methods on which forecasters have previously been reliant no longer satisfy the demands of market participants as they struggle to capture new features associated with today's markets. Hence, this paper investigates two new, contemporary and more proficient methods of forecasting that are capable of identifying and dealing with

discontinuities, nonlinearities and high frequency multi-polynomial components which are all prevalent in financial series of today's markets. These models are most commonly known as the Gene Expression Programming and Genetic Programming Algorithm.

GP Algorithms and GEP are domain-independent problem-solving techniques that are run in various environments. These environments are structured in a manner which approximates problems in order to produce forecasts at a high level of accuracy. GP and GEP can be categorized in the forecasting bracket known in the finance world as 'Evolutionary Algorithms'. The basis for this type of problem – solving techniques derive from the Darwinian principle of reproduction and *survival of the fittest*. Additionally, they are also similar to the biological genetic operations such as crossover recombination and mutation. More importantly, Koza (1990, 1992) underlines that GP and GEP address and quantify complex issues as an automated process via programming, which enable computers to process and solve problems.

The Darwinian aspect of GP and GEP apply the theory of evolution to a population of computer programs of varying sizes and shapes. For instance, GP and GEP start with an initial population of thousands or even millions of randomly generated computer programs. These programs comprise of programmatic elements built to apply the fundamental principles of biological evolution in order to create a new (and often improved) population of programs. As mentioned previously, the creation of this new population is generated in a domain-independent system applying the Darwinian Theory (1859) of natural selection under the principal known as *survival of the fittest*: an analogue of the naturally-occurring genetic operation of sexual recombination (crossover), and occasional mutation. The crossover operation is

designed to create syntactically valid offspring programs. GP combines the expressive high-level symbolic representations of computer programs with the near-optimal efficiency of learning of Holland's (1975) genetic algorithm in order to produce highly accurate outputs. Koza (1998) mentioned that a computer program that solves or at the very least approximates a given problem often emerges from this process. Dissimilar to other models GP do not require any prior knowledge of a model's structure for the purpose of system modelling. Alternatively, GP evolves a system model with parameter values that best fit specific data without manipulating the data to fit 'predefined' model structures as many other forecasting methods tend to do. In other words, GP creates an initial population of models and evolves using genetic operators in order to calculate the mathematical expression which best fits the specified data input into the system. Furthermore, GP simultaneously searches for and refines a model's parameters and ultimately its structure.

On the other hand the GEP process begins with the random generation of the linear chromosomes (or individuals) of the initial population. Then the chromosomes are expressed as Expression Trees (ETs) and the fitness of each individual is evaluated. After that, the individuals are selected according to their fitness in order to be modified by genetic operators and reproduce the new population. The individuals of this new population are, in turn, subjected to the same developmental process: expression of the chromosomes, evaluation, selection according to fitness and reproduction with modification. The process is repeated for a certain number of generations or until a good solution has been found.

Accordingly, the motivation for this chapter is to investigate the use of GP algorithm and GEP in order to improve the forecasting performance using autoregressive

terms as inputs. This is achieved by comparing the two genetic methods with benchmark technical methods such as a moving average convergence/divergence model (MACD), a naïve trading strategy plus an MLP model.

From the analysis it emerges that the GEP algorithm demonstrates a remarkable performance and outperforms all other models in a simple trading simulation exercise. However once more sophisticated trading strategies are utilized with the application of confirmation filters and leverages again the GEP algorithm demonstrates superior forecasting ability in terms of annualised return.

## **7.2 Methodology**

A complete description of our Mixed Neural Network architectures used on this application plus the benchmark models is in chapter 3.

## **7.3 Empirical Trading Simulation Results**

The trading performance of all the models considered in the validation subset is presented in the table below. We choose the model 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. Appendix A.1.8 provides the performance of the GEP and the GP Algorithm in the training and the test sub-periods while Appendices A.1.9, and A.1.6 provide the characteristics of our models and the performance measures respectively. As shown in table 12 below, the ARMA model and the naive strategy have a lower performance compared to the GEP, GP Algorithm and the MLP.

		MLP	GP Algorithm
<b>Information Ratio</b> (excluding costs)		0.60	1.03
<b>Annualised Volatility</b> (excluding costs)		38.11%	38.04%
<b>Annualised Return</b> (excluding costs)		22.99%	39.33%
<b>Maximum Drawdown</b> (excluding costs)		-36.26%	-28.2%
<b>Positions Taken</b> (annualised)		105	67

	NAIVE	MACD	ARMA	Gene Expression
<b>Information Ratio</b> (excluding costs)	0.32	0.46	0.20	1.16
<b>Annualised Volatility</b> (excluding costs)	36.70%	38.12%	38.13%	38.01%
<b>Annualised Return</b> (excluding costs)	11.42%	17.63%	7.68%	44.16%
<b>Maximum Drawdown</b> (excluding costs)	-49.41%	- 50.63%	- 36.50%	-28.20%
<b>Positions Taken</b> (annualised)	119	38	72	71

Table 12: Trading performance results

We can see that the GEP performs significantly better than the GP Algorithm, and the MLP significantly better than the other standard benchmark models despite larger drawdowns.

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.*



(2010), we check for potential improvements to our models through the application of confirmation filters. In our analysis we explored the effect of different thresholds however the results produced were only marginal improvements and for this reason we have decided not to include them in our empirical results.

### 7.3.1 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 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 position.

	MLP	GP Algorithm
<i>Information Ratio (excluding costs)</i>	0.60	1.03
<i>Annualised Volatility (excluding costs)</i>	38.11%	38.04%
<i>Annualised Return (excluding costs)</i>	22.99%	39.33%
<i>Maximum Drawdown (excluding costs)</i>	-36.26%	-28.2%
<i>Positions Taken (annualised)</i>	105	67
<i>Transaction costs</i>	13.65%	9.40%
<i>Annualised Return (including costs)</i>	9.35%	29.93%

	NAIVE	MACD	ARMA	Gene Expression
<i>Information Ratio (excluding costs)</i>	0.32	0.46	0.20	1.16
<i>Annualised Volatility (excluding costs)</i>	36.70%	38.12%	38.13%	38.01%
<i>Annualised Return (excluding costs)</i>	11.42%	17.63%	7.68%	44.16%

<b>Maximum Drawdown (excluding costs)</b>	-49.41%	-	-36.50%	-28.20%
<b>Positions Taken (annualised)</b>	119	38	72	71
<b>Transaction costs</b>	15.47%	4.94%	9.36%	9.94%
<b>Annualised Return (including costs)</b>	-4.05%	12.69%	-1.68%	34.22%

*Table 13: out-of-sample results with transaction costs*

We can see that, after transaction costs, the GEP model outperforms all the other strategies based on the annualized return. It is closely followed by the GP Algorithm strategy. On the other hand, the naïve strategy and the ARMA model produce negative results after transaction costs are taken into account. The MLP achieves decent returns, yet well below those produced by the GP and GEP models.

### 7.3.2 Leverage to Exploit High Sharpe 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%<sup>18</sup> on the test data set.

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. (that is 0.016% per trading day<sup>19</sup>). Our final results are presented in table 14 below.

<sup>18</sup> 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 6.

<sup>19</sup> 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.

	MLP	GP Algorithm
<b>Information Ratio (excluding costs)</b>	0.60	1.03
<b>Annualised Volatility (excluding costs)</b>	40.28%	41.84%
<b>Annualised Return (excluding costs)</b>	24.30%	43.26%
<b>Maximum Drawdown (excluding costs)</b>	-38.32%	-31.02%
<b>Leverage Factor</b>	1.057	1.1
<b>Positions Taken (annualised)</b>	105	67
<b>Transaction and leverage costs</b>	13.65%	9.95%
<b>Annualised Return (including costs)</b>	10.65%	33.31%

	NAIVE	MACD	ARMA	Gene Expression
<b>Information Ratio (excluding costs)</b>	0.32	0.70	0.20	1.16
<b>Annualised Volatility (excluding costs)</b>	36.70%	40.03%	38.13%	39.15%
<b>Annualised Return (excluding costs)</b>	11.42%	18.51%	7.68%	45.48%
<b>Maximum Drawdown (excluding costs)</b>	-49.41%	-53.16%	-36.50%	-29.04%
<b>Leverage Factor</b>	-	1.050	-	1.03
<b>Positions Taken (annualised)</b>	119	38	72	71
<b>Transaction and leverage costs</b>	15.47%	4.94%	9.36%	10.11%
<b>Annualised Return (including costs)</b>	-4.05%	13.57%	-1.68%	35.37%

*Table 14: Trading performance - final results*

As can be seen from table 14, the GEP model continues to demonstrate a superior trading performance despite significant drawdowns. The GP strategy also performs well and presents the second higher annualised returns. In general, we observe that all models are able to gain extra profits, albeit marginally, from the leverage as the increased transaction costs seem to counter most of the benefits. Again it is worth mentioning that the time needed to train the GEP and the GP is almost a quarter of an hour.

## 7.4 CONCLUDING REMARKS

In this chapter, we apply Genetic Programming to a one-day-ahead forecasting and trading task of the ASE 20 fixing series with only autoregressive terms as inputs. We use a naïve strategy, a MACD and an ARMA model as benchmarks. We develop 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 GEP algorithm demonstrates a higher trading performance in terms of annualised return and information ratio before transaction costs and when more elaborate trading strategies are applied. When refined trading strategies are applied and transaction costs are considered the GEP algorithm again continues to outperform all other models achieving the highest annualised return. The GP algorithm model performs remarkably as well and seems to provide good forecasts when autoregressive series are only used as inputs.

It is also important to note that the GP which presents a very close second best performance needs less training time than the GEP algorithm, 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 models and GAs. In particular, the strategies using the GEP and GP algorithms appear quite promising.

## CHAPTER 8

# Stock Market Prediction Using Evolutionary Support Vector Machines: An Application to the ASE20 Index

## Overview

The main motivation for this chapter is to introduce a novel hybrid method for the prediction of the directional movement of financial assets with an application to the ASE20 Greek stock index. Specifically, we use an alternative computational methodology named Evolutionary Support Vector Machine (ESVM) for modeling and trading the ASE20 Greek stock index using as inputs previous values of the ASE20 index and of four other financial indices. The proposed hybrid method consists of a combination of genetic algorithms with support vector machines. For comparison purposes, the trading performance of the ESVM stock predictor is benchmarked with four traditional strategies (a naïve strategy, a Buy and Hold strategy, a MACD and an ARMA models), and a MLP neural network model. As it turns out, the proposed methodology produces a higher trading performance in terms of annualized return and information ratio, while providing information about the relationship between the ASE20 index and other foreign indices.

## 8.1 Introduction

Stock market analysis is an area of growing quantitative financial applications. Modeling and trading financial indices remains nowadays a very challenging open problem for the scientific community. Forecasting financial time series is a difficult task because of their complexity and their nonlinear, dynamic and noisy behaviour.

Traditional methods such as ARMA models and moving average models fail to capture the complexity and the nonlinearities that exist in financial time series. Neural network approaches have given satisfactory results but there is clearly a need for more sophisticated techniques and approaches than the existing ones.

The purpose of this chapter is to present a novel method for the prediction of the directional movement of financial assets with an application to the ASE20 Greek stock index. In order to predict the movement direction of the index our Evolutionary Support Vector Machine (ESVM) model uses as inputs previous values of the ASE20 index and of other financial indices. Therefore, our paper also tries to investigate the relationship between the ASE20, DAX30, Nikkei225, S&P500 and FTSE100 stock market indices.

The chosen methodology designed and developed for predicting the directional movement of the ASE20 index is the ESVM model which is a hybrid method of Genetic Algorithms and Support Vector Machines. Genetic algorithm [14] is an evolutionary heuristic optimization algorithm which has been proved to perform extremely well in practical difficult problems where the search space is big and complicated. Support Vector Machine (SVM) is a supervised learning technique used for data analysis and pattern recognition mainly in classification problems. In our hybrid methodology, a genetic algorithm is used to optimize the SVM parameters and to find the optimal feature subset.

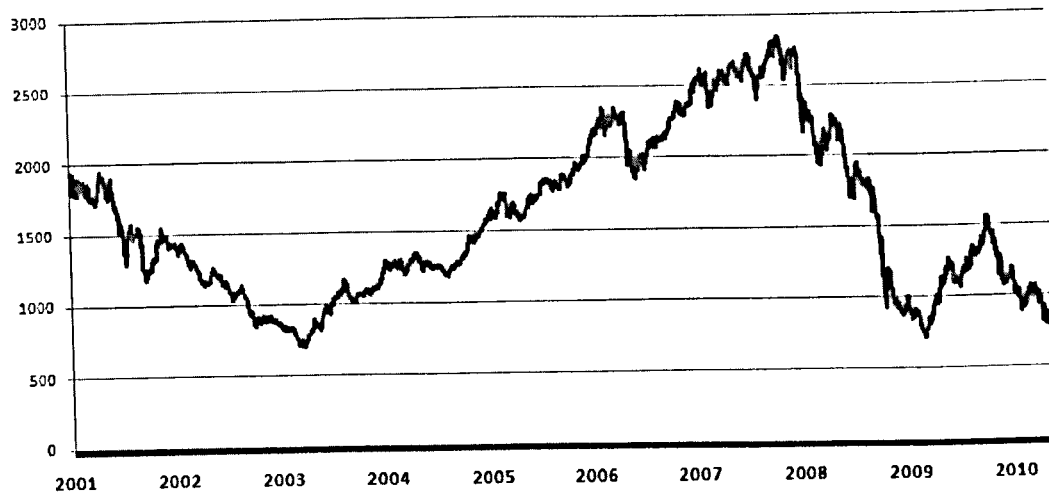
The ESVM stock predictor is benchmarked with four traditional methods (naïve strategy, MACD strategy, ARMA plus a Multilayer Perceptron neural network) and the results obtained seem very promising in terms of annualized return and information ratio.

## 8.2 The ASE20 Index and Related Financial Data

The blue chip index FTSE/ASE20 is the underlying asset for futures contracts on this index that are traded in derivatives markets. The ASE20 index is based on the 20 largest ASE stocks. It was developed in 1997 in partnership between ASE and FTSE International and is now the established benchmark for the Greek stock market. It represents over 50% of ASE's total capitalization and currently has a heavier weight on banking, telecommunication and energy stocks.

The ASE20 index is traded as a futures contract that is cash settled upon maturity of the contract with the value of the index fluctuating on a daily basis. 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. Furthermore, settlement is reached between each of the participating counterparties. Whilst the futures contract is traded in index points the monetary value of the contract is calculated by multiplying the futures price by the multiple of 5 euro per point. For example, a contract trading at 1,400 points is valued at 7,000 EUR.

The ASE20 futures contract is therefore suited to institutional trading which justifies its choice for this empirical application.



*Fig. 18: The ASE-20 Greek stock index*

In order to train our models, we divide our dataset as follows:

<b>Name of Period</b>	<b>Trading Days</b>	<b>Beginning</b>	<b>End</b>
<b><i>Total Dataset</i></b>	2456	1 January 2001	31 May 2010
<b><i>Training Dataset</i></b>	1986	1 January 2001	11 August 2008
<b><i>Validation Set</i></b>	470	12 August 2008	31 May 2010

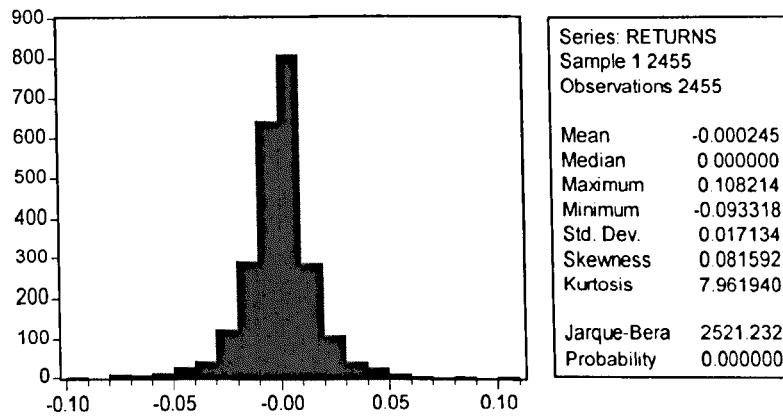
*Table 15: Total dataset*

The observed ASE20 time series is non-normal (Jarque-Bera statistics confirms this at the 99% confidence interval) containing slight skewness and high kurtosis. It is also non-stationary and we decided to transform the ASE20 series into stationary series of rates of return<sup>20</sup>.

Given the price level  $P_1, P_2, \dots, P_t$ , the rate of return at time  $t$  is given by Eq. (25):

<sup>20</sup> The percentage return is linearly additive but the log return is not linearly additive across portfolio components.





*Fig. 19: ASE20 returns summary statistics (total dataset)*

As inputs to our algorithms and our networks, we selected 18 different inputs described in detail in Table 16 below.

Number	Variable	Lag
1	Athens Composite all share return	1
2	Athens Composite all share return	2
3	Athens Composite all share return	3
4	Athens Composite all share return	4
5	Athens Composite all share return	5
6	Athens Composite all share return	6
7	Athens Composite all share return	7
8	Athens Composite all share return	8
9	Athens Composite all share return	9
10	Athens Composite all share return	10
11	Dax30 index return	2
12	10 days moving average of Dax30 index return	2
13	Nikkei 225 index return	1
14	10 days moving average of Nikkei 225 index return	1
15	FTSE100 index return	2
16	10 days moving average of FTSE100 return	2
17	S&P 500 index return	2
18	10 days moving average of S&P 500 return	2

*Table 16: Explanatory variables*

In order to train the ESVM we further divided our dataset as follows:

Name of Period	Trading Days	Beginning	End
<b>Total Dataset</b>	2456	1 January 2001	31 Mai 2010
<b>Training Dataset</b>	1526	1 January 2001	6 November 2006
<b>Test Dataset</b>	460	7 November 2006	11 August 2008
<b>Validation Set</b>	470	12 August 2008	31 Mai 2010

Table 17: SVM combined with Genetic Algorithm Datasets

In Figure 20 we show the Greek stock index for the out-of- sample period.

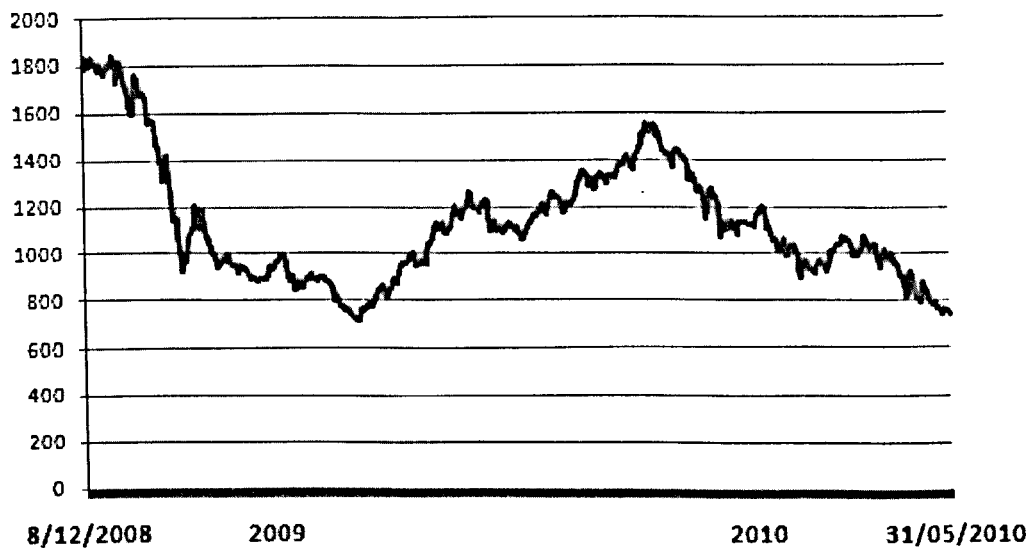


Fig. 20: The ASE-20 Greek stock index (out-of-sample validation period)

As inputs in our forecasting models except from previous values of the ASE20 stock index, we used previous values from other important financial indices. The financial indices which were used are the following:

- **DAX30 Index:** The DAX30 (Deutscher Aktien Index, formerly Deutscher Aktien-Index) is the blue chip stock market index consisting of the 30 largest German companies in terms of order book volume and market capitalization trading on the Frankfurt Stock Exchange. Prices are taken from the electronic Xetra trading system.

- **S&P500 Index:** The Standard & Poor's 500 index is a basket of 500 stocks weighted by market value, and its performance is thought to be representative of the US stock market as a whole. Over 70% of all U.S. equity is tracked by the S&P500 which selects companies based upon market size, liquidity and sector. Most of the companies in the index are solid mid cap or large cap corporations.
- **FTSE100 Index:** The FTSE100 consists of the 100 largest companies by market capitalization on the London Stock Exchange. The composition of the FTSE 100 is reviewed quarterly.
- **NIKKEI225 index:** The Nikkei225 is the benchmark index for the Tokyo Stock Exchange (TSE). It is a price-weighted average (the unit is yen) consisting of 225 Japanese companies, and the components are reviewed once a year. Currently, the NIKKEI225 is the most widely quoted average of Japanese equities.

## 8.2.1 Examining the Relation between the ASE20 Index and other Financial Indices

In order to examine the relationship between the ASE20 index and the other financial indices we first compute the covariance between the ASE20 index values, the value of the previous day's closing price of the other financial index and the value of the moving average for the 10 previous closing prices. In Table 18 we present these results.

<b>Financial Indices and Moving Averages</b>	<b>Pearson correlations (two tailed) with ASE-20 index</b>	<b>Significance level (2-tailed)</b>
DAX30 index	-0.008	0.679

10-day DAX30 Moving Average	0.025	0.210
NIKKEI225 index	-0.035	0.080
10-day NIKKEI225 Moving Average	0.009	0.666
FTSE100 index	-0.008	0.702
10-day FTSE100 Moving Average	0.048	0.019
S&P500	-0.037	0.065
10-day S&P500 Moving Average	0.034	0.092

*Table 18: Pearson correlations of foreign indices with the ASE20 index*

Pearson's correlation measures the linear association between two variables. The values of the correlation coefficient range from -1 to 1. The sign of the correlation coefficient indicates the direction of the relationship (positive or negative). The absolute value of the correlation coefficient indicates the strength, with larger absolute values indicating stronger relationships. The significance level (or p-value) is the probability of obtaining results as extreme as the one observed. If the significance level is very small (for example 0.05) then the correlation is significant and the two variables are linearly related. If the significance level is relatively large (for example, 0.50) then the correlation is not significant and the two variables are not linearly related. We can easily observe that NIKKEI225, S&P500 indices and the 10-day moving averages of FTSE100 and S&P100 indices are according to Pearson's correlation the most linearly correlated with the ASE20 Greek stock index. These conclusions encourage us to continue investigating the impact of foreign indices on the Greek stock market. In order to capture the complex, non-linear relations between foreign financial indices and ASE-20 index, the novel wrapper methodology presented in chapter 3 was applied.

## 8.3 Methodology

A complete description of our Support Vector Machines used on this application plus the benchmark models is in chapter 3.

## 8.4 Empirical Trading Simulation Results

In this section we present the results of the proposed methodology applied to trading the ASE20 Greek stock index. These results are compared with the results of the retained benchmark models. The trading performance of all the models considered in the out-of-sample subset is presented in the table below. Our trading strategy for the proposed methodology is simply the output of the best classifier found. Specifically, we go or stay long if the ESVM model forecasts a positive movement and go or stay short when a negative direction is forecast. The trading strategy applied in benchmark models is simple and identical for all of them: go or stay long when the forecast return is above zero and go or stay short when the forecast return is below zero. Because of the stochastic nature of the proposed methodology a simple run is not enough to measure its performance. This is the reason why ten runs were executed and the mean results are presented in the next tables.

	<b>Buy and Hold</b>	<b>NAIVE</b>	<b>MACD</b>	<b>ARMA</b>	<b>MLP</b>	<b>ESVM</b>
<b><i>Information Ratio (excluding costs)</i></b>	-0.86	1.12	0.13	0.44	1.09	1.65
<b><u>Correct Directional change</u></b>	<b><u>45.11%</u></b>	<b><u>47.02%</u></b>	<b><u>47.45%</u></b>	<b><u>45.96%</u></b>	<b><u>50.00%</u></b>	<b><u>52.77</u></b>
<b><i>Annualised Volatility (excluding costs)</i></b>	44.29%	42.75%	44.35%	44.39 %	44.25%	44.12%
<b><i>Annualised Return (excluding</i></b>	-38.05%	47.78%	5.97%	19.36 %	48.44%	72.71%

<i>costs)</i>						
<b>Maximum Drawdown (excluding costs)</b>	-87.24%	- 32.27%	-46.26%	- 42.15 %	-33.75%	-44.12%
<b>Positions Taken (annualised)</b>	1	120	42	125	84	98

*Table 19: Out of sample trading performance results*

We can see that the ESVM model performs significantly better than the other benchmark methods in terms of information ratio and annualized return. The naïve method outperforms other methods in terms of annualized volatility and maximum drawdown but the differences with the ESVM predictor are insignificant. In terms of positions taken the MACD model and Buy and Hold strategy trade less than all the other methods.

### **8.4.1 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.

We therefore introduce transaction costs as well as leverage for each model. The aim is to devise a trading strategy that takes advantage of the relatively lower volatility of the return profile of some models compared to others.

### **8.4.2 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 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 position.

	<b>NAIVE</b>	<b>MACD</b>	<b>ARMA</b>	<b>MLP</b>	<b>ESVM</b>
<b>Annualised Return (excluding costs)</b>	47.78%	5.97%	19.36%	48.44%	72.71%
<b>Positions Taken (annualised)</b>	120	42	125	84	98
<b>Transaction costs</b>	16.80%	5.88%	17.50%	11.76%	13.72%
<b>Annualised Return (including costs)</b>	30.98%	0.09%	1.86%	36.68%	58.99%

*Table 20: Out-of-sample results with transaction costs*

From Table 20 one can easily see that even when considering transaction costs, the ESVM predictor still significantly outperforms all other benchmark trading strategies in terms of annualized return.

### 8.4.3 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. The leverage factors applied are calculated in such a way that each model has a common volatility of 20%<sup>21</sup> on the test data set.

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%

<sup>21</sup> 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 8.

	<b>NAIVE</b>	<b>MACD</b>	<b>ARMA</b>	<b>MLP</b>	<b>ESVM</b>
<b>Information Ratio (excluding costs)</b>	1.12	0.13	0.44	1.06	1.65
<b>Annualised Volatility (excluding costs)</b>	46.17%	46.57%	48.77%	46.91%	50.74%
<b>Annualized Return (excluding costs)</b>	51.60%	6.27%	21.30%	51.35%	83.62%
<b>Maximum Drawdown (excluding costs)</b>	-34.85%	-49.15%	-46.36%	-35.77%	- 32.02%
<b>Leverage Factor</b>	1.08	1.05	1.10	1.06	1.15
<b>Positions Taken</b>	120	42	125	84	98

p.a. (that is 0.016% per trading day<sup>22</sup>). Our final results are presented in table 21 below.

Table 21 clearly shows that even when considering leverage, the ESVM model still significantly outperforms all other benchmark trading strategies in terms of annualized return.

---

<sup>22</sup> 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.



<i>(annualized)</i>					
<b>Transaction and leverage costs</b>	17.40%	6.25%	18.25%	12.21%	14.84%
<b>Annualized Return (including costs)</b>	34.20%	0.02%	21.11%	39.14%	68.78%

*Table 21: Out-of-sample trading performance - Final results*

From Table 21, it is easily observed that even considering more advance trading techniques like leverage, ESVM predictor still outperforms significantly all other benchmark trading strategies in terms of annualized return and information ratio.

## 8.5 Concluding Remarks

In the present chapter, we introduce a new hybrid methodology which combines genetic algorithms and support vector machines and applies it to the problem of forecasting the next day movement of the ASE20 Greek stock index. For comparative purposes we also apply a naïve trading strategy, a MACD strategy, an ARMA modeling approach and a MLP neural network. Previous values of the ASE20 index and of other important financial indices are used as inputs for our models.

The proposed ESVM methodology produces the highest trading performance in terms of annualized return and information ratio before transaction costs. When leverage and transaction costs are considered, the ESVM model continues to outperform all other benchmark models achieving higher values for the annualized return and information ratio.

It is also important to note that the ESVM methodology was able to uncover relations between the ASE20 stock index and the DAX, the NIKKEI225 and the S&P500 while showing that the FTSE100 index movements do not affect significantly the Greek stock market. These results come in contrast with the results of the linear Pearson

correlations presented in Table 18. One possible explanation for this contradiction is that the information given to our models by the FTSE100 index is probably the same as the information contained in the other selected inputs. Using highly correlated inputs that hold mutual information has been shown to deteriorate the performance of classifiers [21]. Furthermore, simple linear methods like Pearson correlation cannot capture the complex non-linear multiple correlations that exist between the different inputs and only a more sophisticated and powerful technique like GAs can achieve this hard task. The ESVM predictor, by using a wrapper methodology for selecting the optimal feature subset, manages to handle these correlations effectively and this is probably one of the reasons for achieving such promising results.

These obviously need to be confirmed and the application of the ESVM methodology for modeling and trading other financial assets is the next necessary step for our research. Also, the application of the approach proposed by Papadimitriou and Terzidis [22] for deriving a fuzzy rule explanation of our strategy could possibly allow one to more fully understand the impact of every input in the final predictions.

## CHAPTER 9

### General Conclusions

The general motivation of this thesis was to provide empirical evidence on the utility of Combined Neural Networks, Genetic Programming Algorithms, Gene Expression Programming and Support Vector Machines in financial forecasting and trading applications. In order to achieve this, we benchmarked the above models not only with some traditional statistical and technical techniques but also with some other state-of-the-art NNs designs. Therefore, we were able to validate if the theoretical advantages of these techniques compared to the more traditional NNs models are translated into more accurate/profitable forecasts.

In the chapter 4 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 ability in providing good forecasts when autoregressive series are only used as inputs.

Moreover in chapter 5 the Mixed-HONNs demonstrates a higher trading performance in terms of annualised return and information ratio before transaction costs and more elaborate trading strategies are applied. When refined trading strategies are applied and transaction costs are considered the Mixed-MLPs manage to outperform all other models achieving the highest annualised return. The Mixed-HONNs and the

Mixed-RNNs models perform remarkably as well and seem to have an ability in providing good forecasts when autoregressive series are only used as inputs.

In chapter 6 the GP algorithm demonstrates a higher trading performance in terms of annualised return and information ratio before transaction costs. When more elaborate trading strategies are applied and transaction costs are considered the GP algorithm again continues to outperform all other models achieving the highest annualised return. The Mixed-HONNs, the Mixed-RNNs and the Hybrid-HONNs models perform remarkably as well and seem to have ability in providing good forecasts when autoregressive series are only used as inputs.

In chapter 7 the GEP algorithm demonstrates a higher trading performance in terms of annualised return and information ratio before transaction costs and when more elaborate trading strategies are applied. When refined trading strategies are applied and transaction costs are considered the GEP algorithm again continues to outperform all other models achieving the highest annualised return. The GP algorithm model performs remarkably as well and seems to provide good forecasts when autoregressive series are only used as inputs.

Lastly in chapter 8 the proposed ESVM methodology produces the highest trading performance in terms of annualized return and information ratio before transaction costs. When leverage and transaction costs are considered, the ESVM model continues to outperform all other benchmark models achieving higher values for the annualized return and information ratio.

It is also important to note that the ESVM methodology was able to uncover relations between the ASE 20 stock index and the DAX, the NIKKEI225 and the S&P500 while showing that the FTSE100 index movements do not affect significantly the Greek stock market.

The above mentioned empirical evidence allows us to argue with confidence that combined Neural Networks, Genetic Programming Algorithms, Gene Expression Programming and Support Vector Machines can provide accurate and extremely profitable forecasts. Their performance seems superior to that of the HONN MLP and RNN models and of the linear ARMA and MACD techniques. Moreover, we note that the time needed to train Genetic Programming Algorithms, Gene Expression Programming and Support Vector Machines was more than the time needed for RNN MLP and HONN networks. In general, our results should go some way towards convincing quantitative risk and fund managers to use to alternative non-linear techniques such as Genetic Programming Support Vector Machines and Gene Expression Programming as they seem to generate higher return/risk profiles. Finally in few words my findings through my Phd carrier show that the models that i have used in my research gave us promising and remarkable results. Considering the annualised returns,the proposed models are sorted as follows: first the ESVM, second the Gene Expression Programming, third the Genetic Programming Algorithm, fourth the Mixed model and the last is the Hybrid model.

It is worth mentioning that my research in that area will continue with the following papers: 1. Studying the performance of trading models in shock periods. 2. Exploration of interpretable strategies using hybrid evolutionary fuzzy rules methods and 3. Modeling and trading the exchange rates with computational intelligent models.

## REFERENCES

Abraham, A. B., Ajith, A., Baikunth, N. and Mahanti, P. K. (2002), 'Hybrid Intelligent system for Stock Market Analysis', *Computational Science, Springer-Verlag Germany, Vassil n. Alexandrov et al*, 337-345.

Andreou, P. C., Charalambous, C. and Martzoukos, H. S. (2006), 'Knowledge Artificial Neural Networks to Enhanced Parametric Option Pricing', *Research Paper Department of Public and Business Administration, University of Cyprus*.

Barricelli, N. A., (1954), 'Esempi Numerici di Processi di Evoluzione', *Methodos*, 45-68.

Bishop, C., (1994) 'Mixture Density Networks'. *Neural Computing Research Group Report: NCRG/94/004*, 1–25.

Brown, G., Wyatt, J., Harris, R., and Yao, X. (2005) 'Diversity Creation Methods: A Survey and Categorization', *Information Fusion*, 6, 5–20.

Cao L. and Tay F., Support Vector Machine With Adaptive Parameters in Financial Time Series Forecasting, *IEEE Transactions on Neural Networks*, Vol. 14, No. 6, 2003, p.p. 1506 – 1518.

Clemen, R., (1989). 'Combining Forecasts: A Review and Annotated Bibliography', *International Journal of Forecasting*, 5, 559–583.

Cortes C. and Vapnik V. N., Support Vector Networks, *Machine Learning*, Vol. 20, 1995, pp. 1-25.

Cramer, N. L., (1985), 'A Representation for the Adaptive Generation of Simple Sequential Programs', in *Proceedings of an International Conference on Genetic Algorithms and the Applications*, Grefenstette, John J., (ed.), Carnegie Mellon University.

Darwin, C., (1859), '*On the Origin of Species*', London, John Murray.

Dehuri, S. and Cho S. B., (2008), '*Multi-Objective Classification Rule Mining Using Gene Expression*', Third International Conference on Convergence and Hybrid Information.

Dunis, C., Laws, J. and Sermpinis, G. (2008a) 'Higher Order and Recurrent Neural Architectures for Trading the EUR/USD Exchange Rate', *CIBEF Working Papers*. Available at [www.cibef.com](http://www.cibef.com)

Dunis, C., Laws, J. and Sermpinis, G. (2008b) 'Modelling and Trading the EUR/USD Exchange Rate at the ECB Fixing', *CIBEF Working Papers*. Available at [www.cibef.com](http://www.cibef.com)

Dunis, C. and Chen, Y. (2005), 'Alternative Volatility Models for Risk Management and Trading: Application to the EUR/USD and USD/JPY Rates' *Derivative Use, Trading & Regulation*, 11, 2, 126-156

Dunis, C., Laws, j and Karathanasopoulos A. (2010a), 'Modelling and Trading the Greek Stock Market with Hybrid ARMA-Neural Network Models', *CIBEF Working Papers*. Available at [www.cibef.com](http://www.cibef.com)

Dunis, C., Laws, J. and Karathanasopoulos A. (2010), 'GP Algorithm versus Hybrid and Mixed Neural Networks', *CIBEF Working Papers*, Available at [www.cibef.com](http://www.cibef.com).

Elman, J. L. (1990), 'Finding Structure in Time', *Cognitive Science*, 14, 179-211.

Fatima, S. and Hussain, G., (2008) 'Statistical Models of KSE100 Index Using Hybrid Financial Systems', *Neurocomputing*, 7, 2742-2746.

Ferreira C., (2001), 'Gene Expression Programming: A New Adaptive Algorithm for Solving Problems', *Complex Systems*, 13, 87-129.

Ferreira C. (2006), *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*, Springer, 2<sup>nd</sup> edition, San Francisco.

Fogel L. J., Owens A. J., and Walsh, M. J. (1964) 'On the Evolution of Artificial Intelligence', *Proceedings of the Fifth National Symposium on Human Factors in Electronics*, IEEE, San Diego, 63-76.

Fukunaga, A. and Stechert, A. (1998), 'Evolving Nonlinear Predictive Models for Lossless Image Compression with Genetic Programming', *Genetic Programming 1998: Proceedings of the Third Annual Conference*, Morgan Kaufmann Wisconsin, 95 -102.

Giles, L. and Maxwell, T. (1987) 'Learning, Invariance and Generalization in Higher Order Neural Networks', *Applied Optics*, 26, 4972-4978.

Greg, T. and Hu, S. (1999), 'Forecasting GDP Growth Using Artificial Neural Network', *Working Paper, Bank of Canada*, 99-3.

Hansen, J. and Nelson, R., (2003) 'Time-Series Analysis with Neural Networks and ARIMA-Neural Network Hybrids', *Journal of Experimental and Theoretical Artificial Intelligence*, 15 (3), 315-330.

- Hibbert, H., Pedreira, C. and Souza, R., (2000) 'Combining Neural Networks and ARIMA Models for Hourly Temperature Forecast', *Proceedings of International Conference on Neural Networks (IJCNN 2000)*, 414–419.
- Hibon, M. and Evgeniou. T., (2005) 'To Combine or not to Combine: Selecting among Forecasts and their Combinations', *International Journal of Forecasting*, 22, 15-24.
- Holland J.H. (1975), *Adaptation in Natural and Artificial System*, The University of Michigan Press, Ann Arbor.
- Holland J., *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, Cambridge, Mass: MIT Press, 1995.
- Huang W., Nakamori Y. and Wang S., Forecasting Stock Market Movement Direction With Support Vector Machine, *Computers & Operations Research*, Vol. 32, 2005, pp. 2513–2522.
- Ince H. and Trafalis T., Short Term Forecasting with Support Vector Machines and Application to Stock Price Prediction, *International Journal of General Systems*, Vol. 37, No. 6, 2008, pp. 677–687.
- Kaastra, I. and Boyd, M. (1996), 'Designing a Neural Network for Forecasting Financial and Economic Time Series', *Neurocomputing*, 10, 215-236.
- Kerthi, S. and Lin, C. J., Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel, *Neural Computation*, Vol. 15, 2003, pp. 1667-1689.
- Kim K., Financial Time Series Forecasting Using Support Vector Machines, *Neurocomputing*, Vol. 55, 2003, pp. 307-319.
- Knowles, A., Hussein, A., Deredy, W., Lisboa, P. and Dunis, C. L. (2009), 'Higher-Order Neural Networks with Bayesian Confidence Measure for Prediction of EUR/USD Exchange Rate', *Artificial Higher Order Neural networks for Economic and Business*, 1, 48-59, *CIBEF Working Papers*. Available at [www.cibef.com](http://www.cibef.com).
- Koza, J.R. (1990), 'Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems', *Stanford University Computer Science Department*.
- Koza, J.R. (1992), *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge MIT Press
- Koza, J.R. (1994), *Genetic Programming II: Automatic Discovery of Reusable Programs*, Cambridge MIT Press



Koza J.R. (1998), 'Genetic Programming', In Williams, J. G. and Kent, A., (eds.), *Encyclopedia of Computer Science and Technology*. New York, NY: Marcel-Dekker. 39, (Supplement 24), 29–43.

Koza, J.R., Bennett, F.H., Andre, D. and Keane, M.A., (1999), *Genetic Programming III: Darwinian Invention and Problem Solving*, San Francisco, Morgan Kaufmann.

Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J. and Lanza, G. (2003), *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*, Kluwer Academic Publishers.

Lindemann, A., Dunis, C, and Lisboa P. (2004), 'Level Estimation, Classification and Probability Distribution Architectures for Trading the EUR/USD Exchange Rate'. *Neural Network Computing & Applications*, 14, 3, 256-271.

Lisboa, P. J. G. and Vellido, A. (2000), 'Business Applications of Neural Networks', vii-xxii, in P. J. G. Lisboa, B. Edisbury and A. Vellido [eds.] *Business Applications of Neural Networks: The State-of-the-Art of Real-World Applications*, World Scientific, Singapore.

Lopez, H. S. and Weinert, W. R. (2004), 'An Enhanced Gene Expression Programming Approach for Symbolic Regression Problems', *International Journal of Applied Mathematics in Computer Science*, 14, 375-384.

Madár, J., Abonyi, F. and Szeifert, F. (2005) 'Genetic Programming for the Identification of Nonlinear Input-Output Models', Industrial and Engineering Chemistry Research. p.o Box 158 Veszprem 8201 Hungary

Madár, J., Abonyi, F. and Szeifert, F. (2004) 'Genetic Programming for System Identification', *Intelligent Systems Design and Applications* (ISDA). University of Veszprem, Hungary

Makridakis, S., Anderson A., Carbone, R., Fildes, R., Hibdon, M., Lewandowski, R., Newton, J., Parzen, E. and Winkler, R., (1982) 'The Accuracy of Extrapolation (Time Series) Methods: Results of a Forecasting Competition', *Journal of Forecasting*, 1, 111–153.

Makridakis, S., (1989) 'Why Combining Works?', *International Journal of Forecasting*, 5, 601–603.

Margny, M. H. and El-Semman I. E., (2005) 'Extracting Logical Classification Rules with Expression Programming: Micro Array Case Study', AIML 05 , Conference 19-21 December , Cairo, Egypt.

Newbold, P. and Granger, C. W. J., (1974) 'Experience with Forecasting Univariate Time Series and the Combination of Forecasts (with discussion)', *Journal of Statistics*, 137, 131–164.

- Palm, F.C. and Zellner, A., (1992) 'To Combine or not to Combine? Issues of Combining Forecasts', *Journal of Forecasting*, 11, 687–701.
- Papadimitriou, S. and Terzidis, K., Efficient and Interpretable Fuzzy Classifiers from Data with Support Vector Learning, *Intelligent Data Analysis*, Vol. 9, 2005, pp. 527-550.
- Pindyck, R. and Rubinfeld, D. (1998), *Econometric Models and Economic Forecasts*, 4<sup>th</sup> edition, McGraw-Hill, New York.
- Rechenberg, I. (1971), 'Evolutionsstrategie - Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution', (*PhD thesis*). Reprinted by Fromman-Holzboog (1973).
- Redding, N., Kowalczyk, A. and Downs, T. (1993), 'Constructive Higher-Order Network Algorithm that is Polynomial Time', *Neural Networks*, 6, 997-1010.
- Shapiro, A. F. (2000), 'A Hitchhiker's Guide to the Techniques of Adaptive Nonlinear Models', *Insurance, Mathematics and Economics*, 26, 119-132.
- Scholkopf B., Mika S., Burges J. C., Knirsch P., Muller K.-R., Ratsch G. and Smola A., Input Space Versus Feature Space In Kernel-Based Methods, *IEEE Transactions on Neural Networks*, Vol. 10, No. 5, 1999, p.p. 1000 - 1017.
- Scholkopf B. and Smola A. J., *Learning with Kernels: Support Vector Machines, Regularization and Beyond*, Cambridge, Mass: MIT Press, 2002.
- Tenti, P. (1996), 'Forecasting Foreign Exchange Rates Using Recurrent Neural Networks', *Applied Artificial Intelligence*, 10, 567-581.
- Terui, N. and van Dijk, H. (2002) 'Combined Forecasts from Linear and Nonlinear Time Series Models', *International Journal of Forecasting*, 18, 421–438.
- Theil, H (1996), 'Applied Economic Forecasting, North-Holland, Amsterdam, Netherlands.
- Tsang, E. P. K., Butler J. M. and Li, J. (1998), 'EDDIE Beats the Bookies', *Journal of Software Practice and Experience*, Wiley, 28, (10), 1033-1043.
- Tseng, F.M., Yu, H.C. and Tzeng, G.H., (2002) 'Combining Neural Network Model with Seasonal Time Series ARIMA Model', *Technological Forecasting and Social Change*, 69, 71–87.
- Vapnik V. N., *The Nature of Statistical Learning Theory*, Springer, 2000, United States of America.
- Wang, Y. F., (2007) 'Nonlinear Neural Network Forecasting Model for Stock Index Option Price: Hybrid GJR-GARCH Approach', *Expert Systems with Applications*, 475-484.

Werner, J. C. and Fogarty, T. C. (2001), 'Genetic Programming Applied to Collagen Disease & Thrombosis', *South Bank University*, London

Willis, M.J., Hiden, H. G., Marenbach, P., McKay, B, and Montague, G. A.(1997), 'Genetic Programming: An Introduction and Survey of Applications', *Second International Conference on Genetic Algorithms in Engineering Systems*, 314 – 319.

Winkler, S. (2004), 'Identifying Nonlinear Model Structures Using Genetic Programming'. *Diploma Thesis, Institute of Systems Theory and Simulation, Johannes Kepler University Linz, Austria.*

Winkler, S., Affenzeller, M. and Wagner, S. (2004a), 'New Methods for the Identification of Nonlinear Model Structures Based Upon Genetic Programming Techniques', *Proceedings of the 15th International Conference on Systems Science*, 1, 386-393.

Winkler, S., Affenzeller, M. and Wagner, S. (2004b), 'Identifying Nonlinear Model Structures Using Genetic Programming Techniques', *Cybernetics and Systems*, 689-694.

Zhang, M., Xu, S., X. and Fulcher, J. (2002), 'Neuron-Adaptive Higher Order Neural-Network Models for Automated Financial Data Modelling', *IEEE Transactions on Neural Networks*, 13, 1, 188-204.

Zhang, G.P., (2003) 'Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model', *Neurocomputing*, 50, 159–175.

Zhang, G. P., and Qi, M., (2005) 'Neural Network Forecasting for Seasonal and Trend Time Series', *European Journal of Operational Research*, 160 (2), 501–514.

# APPENDIX

## A.1.1 ARMA Model

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

Dependent Variable: RETURNS  
 Method: Least Squares  
 Date: 03/17/09 Time: 22:18  
 Sample (adjusted): 8 1738  
 Included observations: 1731 after adjustments  
 Convergence achieved after 37 iterations  
 Backcast: 1 7

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	1.856760	Prob(F-statistic)		0.000000
Inverted AR Roots	.89-.44i	.89+.44i	.31-.92i	.31+.92i
	-.54+.70i	-.54-.70i	-.93	
Inverted MA Roots	.88-.45i	.88+.45i	.31-.92i	.31+.92i
	-.54+.70i	-.54-.70i	-.94	

*Table 22: The ARMA model benchmark*

### A.1.2 Explanatory variables for our Models

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 23: 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	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

*Table 24: Explanatory variables for Hybrid Neural Networks*

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

*Table 25: Explanatory variables for Mixed Neural Networks*

Number	Variable	Lag
1	Athens Composite all share return N	1
2	Athens Composite all share return N	3
3	Athens Composite all share return N	5
4	Athens Composite all share return N	7
5	Athens Composite all share return N	8
6	Athens Composite all share return N	10
7	Dax30 index return N	2
8	10 days moving average of Dax30 index return N	2
9	10 days moving average of Nikkei 225 index return N	1
10	S&P 500 index return N	2
11	10 days moving average of S&P 500 return N	2

*Table 26: Input variables selected by ESVM model*

### A.1.3 Data Segregation

Name of Period	Trading Days	Beginning	End
<i>Total Dataset</i>	2087	21 January 2001	31 December 2008
<i>Training Dataset</i>	1719	29 January 2001	30 August 2007
<i>Out- of- sample Dataset(Validation Set)</i>	349	31 August /2007	31 December 2008

*Table 27: The ASE 20 dataset*

2087 21 January 2001 31 December 2008

<b>Total Dataset</b>			
<b>Training Dataset</b>	1373	29 January 2001	03 May 2006
<b>Test Dataset</b>	346	04 May 2006	30 August 2007
<b>Out-of- sample Dataset (Validation Set)</b>	349	31 August 2007	31 December 2008

Table 28: The Neural Networks datasets

### A1.4 Graph of Entire Dataset

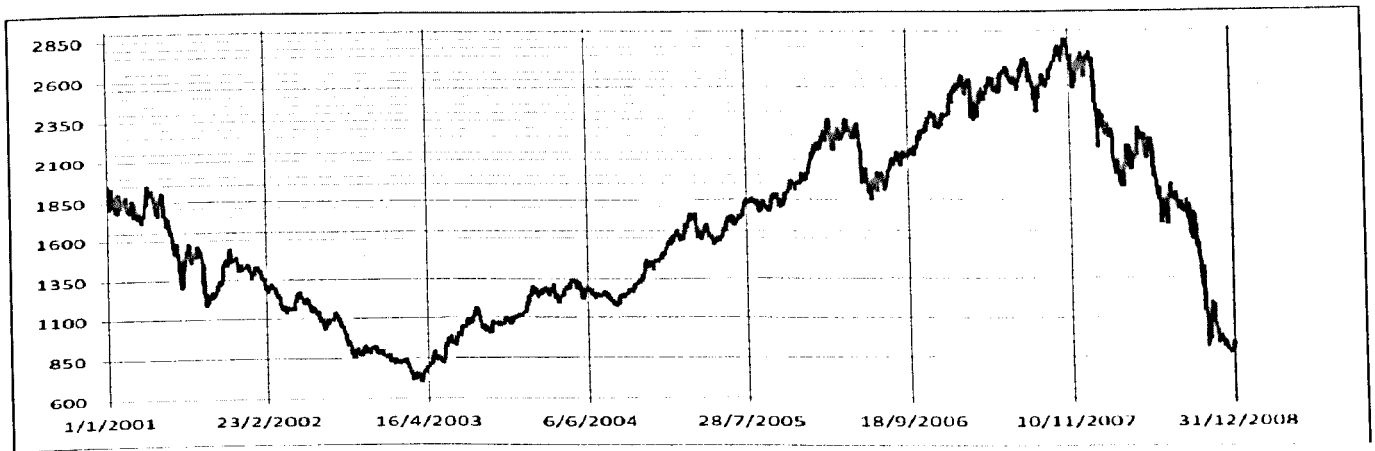


Fig. 21: ASE 20 fixing prices (total dataset).

### A.1.5 Histogram of Returns

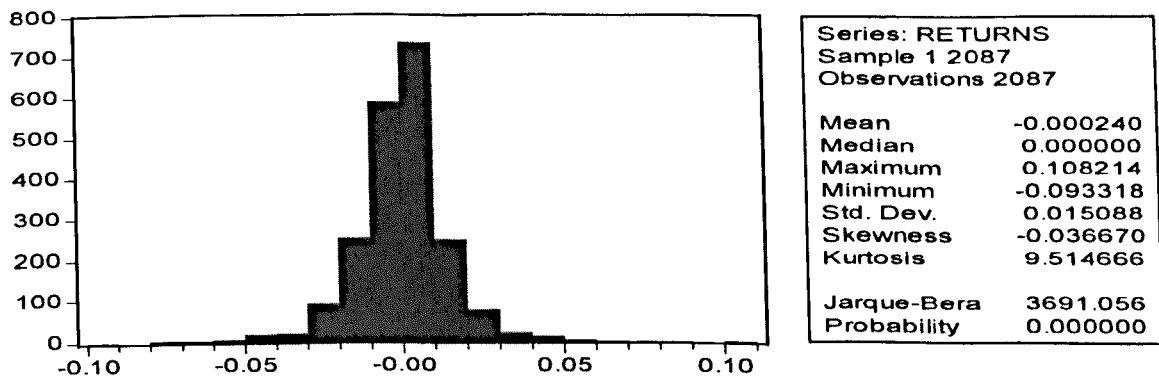


Fig. 22: ASE 20 returns summary statistics (total dataset).

### A.1.6 Performance Measures

The performance measures are calculated as follows:

Performance Measure	Description
Annualised Return	$R^A = 252 * \frac{1}{N} \sum_{i=1}^N R_i$ <p>with <math>R_i</math> being the daily return</p>
Cumulative Return	$R^C = \sum_{i=1}^N R_i$
Annualised Volatility	$\sigma^A = \sqrt{252} * \sqrt{\frac{1}{N-1} * \sum_{i=1}^N (R_i - \bar{R})^2}$
Information Ratio	$IR = \frac{R^A}{\sigma^A}$
Maximum Drawdown	<p>Maximum negative value of <math>\sum (R_i)</math> over the period</p> $MD = \text{Min}_{i=1, \dots, T, T=1, \dots, N} \left( \sum_{j=1}^i R_j \right)$

*Table 29: Trading simulation performance measures*

### A.1.7 Statistical Results in the Training and Test Sub-Periods

	NAIVE	MACD	ARMA	MLP	RNN	HONN
RMSE	0.0125	0.0131	0.0124	0.0153	0.0237	0.0141
MAE	0.0125	0.0097	0.0090	0.0111	0.0119	0.0103
MAPE	456.56%	235.17%	117.82%	371.57%	329.88%	234.72%
THEIL-U	0.6781	0.7459	0.8643	0.6842	0.7174	0.6938

	Hybrid-MLP	Hybrid-RNN	Hybrid-HONN
RMSE	0.0118	0.0122	0.0118
MAE	0.0086	0.0082	0.0081
MAPE	108.93%	128.02%	124.91%
THEIL-U	0.7226	0.776	0.6862

	Mixed MLP	Mixed RNN	Mixed HONN
RMSE	0.0127	0.0205	0.0189
MAE	0.0091	0.0116	0.0126
MAPE	111.34%	285.99%	355.95%
THEIL-U	0.7881	0.7105	0.6989

*Table 30: In-sample statistical performance*



## A.1.8 Empirical Results in the Training and Test Sub-Periods

	NAIVE	MACD	ARMA	MLP	RNN	HONN
<i>Information Ratio (excluding costs)</i>	1.55	1.24	1.24	1.57	1.53	1.61
<i>Annualised Volatility (excluding costs)</i>	19.32%	19.49%	19.83%	19.60%	19.60%	19.59%
<i>Annualised Return (excluding costs)</i>	29.86%	24.29%	24.66%	30.72%	30.02%	31.56%
<i>Maximum Drawdown (excluding costs)</i>	-23.39%	-25.42%	-26.70%	-27.52%	-34.66%	-39.70%
<i>Positions Taken (annualised)</i>	114	34	50	86	81	108

	Hybrid-MLP	Hybrid-RNN	Hybrid-HONN
<i>Information Ratio (excluding costs)</i>	2.13	2.01	2.26
<i>Annualised Volatility (excluding costs)</i>	19.42%	19.44%	19.40%
<i>Annualised Return (excluding costs)</i>	41.35%	39.01%	43.77%
<i>Maximum Drawdown (excluding costs)</i>	-37.20%	-26.86%	-37.20%
<i>Positions Taken (annualised)</i>	102	79	77

	Mixed-MLP	Mixed-RNN	Mixed-HONN
<i>Information Ratio (excluding costs)</i>	2.07	1.93	2.11
<i>Annualised Volatility (excluding costs)</i>	19.45%	19.47%	19.44%
<i>Annualised Return (excluding costs)</i>	40.17%	37.57%	41.12%
<i>Maximum Drawdown (excluding costs)</i>	-37.89%	-41.47%	-37.52
<i>Positions Taken (annualised)</i>	46	68	47

	MLP	GP	GEP
<i>Information Ratio (excluding costs)</i>	0.60	2.19	2.34
<i>Annualised Volatility (excluding costs)</i>	38.11%	19.33%	19.31%
<i>Annualised Return (excluding costs)</i>	22.99%	42.24%	45.19%
<i>Maximum Drawdown (excluding costs)</i>	-36.26%	-31.23%	-28.07%
<i>Positions Taken (annualised)</i>		50	52

*Table 31: In-sample trading performance*

## A.1.9 Networks Characteristics

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

Parameters	MLP	RNN	HONNs
<i>Learning algorithm</i>	<i>Gradient descent</i>	<i>Gradient descent</i>	<i>Gradient descent</i>
<i>Learning rate</i>	0.001	0.001	0.001
<i>Momentum</i>	0.003	0.003	0.003
<i>Iteration steps</i>	1500	1500	1000
<i>Initialisation of weights</i>	N(0,1)	N(0,1)	N(0,1)
<i>Input nodes</i>	11	11	11
<i>Hidden nodes (1layer)</i>	7	6	0
<i>Output node</i>	1	1	1

*Table 32: Network Characteristics for Traditional Neural Networks*

Parameters	Hybrid-MLP	Hybrid-RNN	Hybrid-HONNs
<b>Learning algorithm</b>	<b>Gradient descent</b>	<b>Gradient descent</b>	<b>Gradient descent</b>
<b>Learning rate</b>	0.001	0.001	0.001
<b>Momentum</b>	0.003	0.003	0.003
<b>Iteration steps</b>	1500	1500	1000
<b>Initialisation of weights</b>	$N(0,1)$	$N(0,1)$	$N(0,1)$
<b>Input nodes</b>	13	13	13
<b>Hidden nodes (1layer)</b>	6	7	0
<b>Output node</b>	1	1	1

*Table 33: Network characteristics for Hybrid Neural Networks*

Parameters	Mixed-MLP	Mixed-RNN	Mixed-HONN
<b>Learning algorithm</b>	<b>Gradient descent</b>	<b>Gradient descent</b>	<b>Gradient descent</b>
<b>Learning rate</b>	0.001	0.001	0.001
<b>Momentum</b>	0.003	0.003	0.003
<b>Iteration steps</b>	1500	1500	1000
<b>Initialisation of weights</b>	$N(0,1)$	$N(0,1)$	$N(0,1)$
<b>Input nodes</b>	13	13	13
<b>Hidden nodes (1layer)</b>	6	7	0
<b>Output node</b>	1	1	1

*Table 34: Network characteristics for Mixed Neural Network*

### A.1.10 Genetic Programming Characteristics

We present below the characteristics of the Genetic Programming Algorithm with the best trading performance on the test sub-period.

<b>Population Size:</b>	200
<b>Max tree depth:</b>	6
<b>Function Set:</b>	+, -, *, /, ^, ^2, ^3, ^1/2, ^1/3, Exp, If, sin, cos, tan
<b>Fitness evaluation function:</b>	Mean Squared Error
<b>Tournament Size:</b>	4
<b>Crossover trials:</b>	1
<b>Mutation Probability:</b>	0,75

*Table 43: Genetic Programming characteristics*

<b>Population Size:</b>	1000
<b>Head length:</b>	6
<b>Constants' range:</b>	[-3, 3]
<b>Function Set:</b>	+, -, *, /, ^, ^2, ^3, ^1/2, ^1/3, Exp, If, sin, cos, tan
<b>Fitness evaluation function:</b>	Mean Squared Error
<b>Tournament Size:</b>	20
<b>Type of recombination:</b>	Two point
<b>Mutation Probability:</b>	0,75

*Table 35: Gene Expression Programming Characteristics*

## A.2.1 ARMA Model

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

Dependent Variable: RETURNS  
 Method: Least Squares  
 Date: 10/18/10 Time: 13:23  
 Sample (adjusted): 22 1895  
 Included observations: 1874 after adjustments  
 Convergence achieved after 14 iterations  
 Backcast: 1 21

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	0.000173	0.000290	0.594784	0.5521
AR(2)	-0.048839	0.009521	-5.129866	0.0000
AR(7)	-1.201091	0.047436	-25.32030	0.0000
AR(15)	0.028249	0.005333	5.296766	0.0000
AR(21)	0.260213	0.040526	6.420878	0.0000
MA(2)	0.062896	0.011921	5.276268	0.0000
MA(7)	1.188975	0.047036	25.27820	0.0000
MA(15)	-0.036059	0.007143	-5.048213	0.0000
MA(21)	-0.263141	0.038319	-6.867106	0.0000
R-squared	0.032286	Mean dependent var		0.000176
Adjusted R-squared	0.028135	S.D. dependent var		0.012792
S.E. of regression	0.012611	Akaike info criterion		-5.903711
Sum squared resid	0.296602	Schwarz criterion		-5.877125
Log likelihood	5540.777	F-statistic		7.777859
Durbin-Watson stat	1.876937	Prob(F-statistic)		0.000000
Inverted AR Roots	.88	.88+.44i	.88-.44i	.86-.41i
	.86+.41i	.55-.69i	.55+.69i	.22+.97i
	.22-.97i	.20-.92i	.20+.92i	-.19+.86i
	-.19-.86i	-.60+.79i	-.60-.79i	-.61-.73i
	-.61+.73i	-.79+.38i	-.79-.38i	-.97+.04i
	-.97-.04i			
Inverted MA Roots	.88	.87+.44i	.87-.44i	.87+.40i
	.87-.40i	.55+.69i	.55-.69i	.23+.97i
	.23-.97i	.20+.93i	.20-.93i	-.19-.86i
	-.19+.86i	-.59-.80i	-.59+.80i	-.62+.72i
	-.62-.72i	-.79+.38i	-.79-.38i	-.97+.04i
	-.97-.04i			

*Table 36: The ARMA model benchmark*

## A.2.2 Empirical Results in the Training and Test Sub-Periods

	MLP	ESVM
<i>Information Ratio (excluding costs)</i>	2.41	1.79
<i>Annualised Volatility (excluding costs)</i>	20.78%	20.89%
<i>Annualised Return (excluding costs)</i>	50.01%	37.38%
<i>Maximum Drawdown (excluding costs)</i>	-25.25%	-20.82%

	NAIVE	MACD	ARMA	Buy & Hold
<i>Information Ratio (excluding costs)</i>	1.20	1.05	1.58	0.11
<i>Annualised Volatility (excluding costs)</i>	20.45%	20.86%	20.88%	21.12%
<i>Annualised Return (excluding costs)</i>	24.60%	21.96%	32.93%	2.36%
<i>Maximum Drawdown (excluding costs)</i>	-38.87%	-26.27%	-23.47%	-96.35%

*Table 37: In-sample trading performance*

## A.2.3 ESVM Characteristics

<b>Population Size:</b>	30
<b>Selection type:</b>	Roulette Wheel Selection
<b>Elitism:</b>	Best member of every population is maintained in the next generation
<b>Crossover Probability:</b>	0.9
<b>Mutation Probability:</b>	0.1

*Table 38: ESVM Parameters*

## A.2.4 Networks Characteristics

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

Parameters	MLP
<i>Learning algorithm</i>	<i>Gradient descent</i>
<i>Learning rate</i>	0.001
<i>Momentum</i>	0.003
<i>Iteration steps</i>	1500
<i>Initialisation of weights</i>	N(0,1)
<i>Input nodes</i>	18
<i>Hidden nodes (1layer)</i>	9
<i>Output node</i>	1

*Table 39: Characteristics for Multilayer Perceptron*

### **A.2.5 Abbreviation list**

ANN = Artificial Neural Network

ARMA = Autoregressive Moving Average

ESVM = Evolutionary Support Vector Machine

GA = Genetic Algorithm

GEP = Gene Expression Programming

GP = Genetic Programming

HONN = Higher Order Neural Network

MA = Moving Average

MLP = Multilayer Perceptron

RNN = Recurrent Neural Network