Expert Systems with Applications 27 (2004) 417–425

Expert Systems with Applications

www.elsevier.com/locate/eswa

Predicting the exchange traded fund DIA with a combination of genetic algorithms and neural networks

Massimiliano Versace*, Rushi Bhatt[1], Oliver Hinds[2], Mark Shiffer[3]

Department of Cognitive and Neural Systems, Boston University, Boston, MA 02215, USA

Abstract

We evaluate the performance of a heterogeneous mixture of neural network algorithms for predicting the exchange-traded fund DIA. A genetic algorithm is utilized to find the best mixture of neural networks, the topology of individual networks in the ensemble, and to determine the features set. The genetic algorithm also determines the window size of the input time-series supplied to the individual classifiers in the mixture of experts. The mixtures of neural network experts consist of recurrent back-propagation networks, and radial basis function networks. The application of genetic algorithm on the heterogeneous mixture of powerful neural network architectures shows promise for prediction of stock market time series. These highly non-linear, stochastic and highly non-stationary time series have been found to be notoriously difficult to predict using conventional linear statistical methods. In this paper, we propose a biologically inspired methodology to tackle such hard problem using a multi-faceted solution.

Keywords: Time series prediction; Financial forecasting; Genetic algorithms; Neural networks

1. Introduction

During the last few years, financial institutions as well as individual investors have found a wide range of uses for Artificial Intelligence (AI) technologies such as expert systems, artificial neural networks (ANNs), genetic algorithms (GAs), and fuzzy logic. Financial engineering has therefore become the natural theatre where pattern classification algorithms can be contrasted and tested on highly demanding grounds. The main rationale motivating the use of these techniques is to identify and exploit the regularity that is hidden in the apparently chaotic price trend of a given security. Parallel to the development if AI technologies, the last decade has also witnessed the development of several nonlinear time series models (Granger & Anderson, 1978, Tong & Lim, 1980, Engle, 1982). These nonlinear models, although not suffering from a priori assumption of a linear relationship between data and time series, are limited by the fact that an a priori assumption of the nature of the nonlinear relationship must be formulated, the latter task being even more demanding than the linear case.

Within the wider field of AI, ANNs have been demonstrated to be a natural solution to problems where an explicit description of the nature of the data is not available due to their capability to generate data-driven hypotheses. Historically, the development of these algorithms was inspired by investigations into the functioning of the nervous system. ANNs have subsequently been found to have sound theoretical basis from the perspective of statistical learning theory, and they usually yield good performance when used for real-world data analysis or in predicting nonlinear dynamical systems (Lapedes & Farber, 1987, 1988; Haykin, 1999; Duda, Hart, & Stork, 2000). ANNs have good generalization capabilities and are usually robust against noisy or missing data, all of which are highly desirable properties time series prediction. Importantly, this class of unsupervised neural networks can operate a much richer family of functions compared to linear regression based techniques. Therefore, unsupervised neural networks have the capability of discovering associations between features that may not have been expected or looked for.

There is an extensive literature on financial applications of ANNs (Trippi & Turban, 1993; Azoff, 1994; Refenes, 1995; Gately, 1996; Odom & Sharda, 1990; Coleman, Graettinger, & Lawrence 1991; Salchenkerger, Vinar, & Lash, 1992;

* Corresponding author. Tel.: +617-353-6426; fax: +617-353-7755.
E-mail addresses: vsrsace@cns.bu.edu (M. Versace), rushi@bu.edu (R. Bhatt), oph@bu.edu (O. Hinds), mshiffer@cns.edu (M. Shiffer).

[1] Tel.: +617-353-5235; fax: +617-353-7755.
[2] Tel.: +617-353-6426; fax: +617-353-7755.
[3] Tel.: +617-353-6181; fax: +617-353-7755.

0957-4174/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2004.05.018

Tam & Kiang, 1992; Wilson & Sharda, 1994, Weigend, Rumelhart, & Hubermann, 1992; Zhang, 1998, for a review). Although encouraging results have been reported in which ANNs-based systems outperformed widely-used well-established statistical methods, many inconsistent reports have been undermining the robustness of these findings. Among the reasons of these discrepancies are well-known problems that characterize ANNs, in particular:

(1) Different network type (linear filters, multilayer perceptrons, radial basis functions networks, or RBF, and self-organizing maps, among others) can lead to different results when trained and tested on the same database. This is mainly due to the different classes of decision boundaries that different ANN types prefer;
(2) For a given network type, ANNs are sensitive to the choice of topology and size for a given data set;
(3) ANNs are prone to overfitting, unless great care is taken in choosing the size and connectivity of the network;
(4) The highly variable nature of financial time series can prevent a single ANN from producing accurate forecasts for an extended trading period, even thought it can perform well above chance in a given testing set.

Combining classifiers and boosting methods often lead to improvement in performance over single neural networks. Several studies (Pelikan, de Groot, & Wurtz, 1992; Ginzburg & Horn, 1994; Zhang, 1994; Chu & Widjaja, 1994) have shown improvement of performance in combining different ANNs or training similar ANNs on different features of the input data and then recombining their output in a later stage. Our usage of a mixture of experts is motivated by the hypothesis that a particularly difficult problem can be broken into smaller sub-problems which are easier to solve with a single classifier. This *divide et impera* principle, so common in computer science, is particularly suited for difficult problems, like modelling of financial time series. The mixture of experts' framework specifies that a prediction is made up of a series of predictions from separate experts, or networks. In general, the final output is decided by a function that combines individual classification outputs:

$$O = f(\vec{y}) \tag{1}$$

where $\vec{y}$ is a vector of individual classifier output, and O is the output of the mixture of experts. A common implementation of a mixture of expert uses a weighted combination of y_i :

$$\hat{y} = \sum_{i=1}^{m} g_i \hat{y}_i \tag{2}$$

where m is the number of experts in the model, $\hat{y}_i$ is the prediction of the expert i and g_i is the weighting on expert i. This weighting can itself be the prediction of another model, known as the gate, or can simply be set to 1 for each classifier. The results of the application of mixture of experts are encouraging, with these systems often outperforming standard statistical techniques and other classification methods, although it is difficult to say in general in what kind of tasks mixtures-of-experts or any other kind of model will perform well on any given dataset (Kang & Oh, 1997, 2000; Jordan & Jacobs, 1994; Jordan & Xu, 1995; Waterhouse, 2002).

In the ANNs literature, inappropriate topology selection and weight training are frequently blamed for poor performance. Increasing the number of hidden layer neurons helps improving network performance, yet many problems could be solved with very few neurons if only the network took its optimal configuration. Due to the large size of the hypothesis space that these ANNs are capable of generating, it is difficult to guarantee that an ANN will converge to a globally optimal hypothesis for the distribution of the underlying the given dataset.

The inherent nonlinearity of ANNs results in the existence of many sub-optimal networks, and the great majority of training algorithms converge to these sub-optimal configurations (local minima). Solutions to local minima problems often imply methods that are probabilistic in their nature. The methods can in fact find the globally optimal solution with a certain probability, which usually depends on the number of iterations of the algorithm. At the same time, the danger of overfitting is always present, rendering the problem of the solution of an optimal network configuration very difficult.

To summarize, there are multiple factors that influence the choice of a given system of networks, and within a given network type there are multiple combinations of parameters, network architecture, activation and learning functions, input selection and preprocessing that produce a combinatorial explosion of possible systems.

A natural solution for searching this very large space of system configuration is provided by GAs. GAs are a class of probabilistic search techniques that has been developed in the past decades as a general-purpose optimization tool (Holland, 1975; Goldberg, 1989). GAs mimic biological evolution by using a massively parallel search mechanism that involves (a) the initialization of a random population of systems (or candidate solutions) (b) ordering the systems based on a measure of success in approximating the desired solution (fitness) (c) a reproduction stage, in which the best exemplars of the last generation have the chance to produce an offspring through the application of some GA operators, namely crossover and mutation. GAs are suited for particularly hard problems when little or no knowledge of the optimal function is given and the search space is very large. GAs are thus an ideal tool for solving the problem if discovering appropriate parameterizations for ANNs, and good results have been obtained in combining GAs and ANNs in hybrid systems (Belew, McInnernay, & Schraudolph, 1990; Montana & Davis, 1989; Shaffer, Whitely, & Eshelman, 1990; Chang & Lippmann, 1991; Harp & Samad, 1991; Miller, Todd, & Hedge, 1989; Whitley, 1989; Kingdon, 1997; Venkatesan & Kumar, 2002).

This paper presents an application of such a hybrid system that uses GAs for selecting an appropriate combination of networks, parameters and training regimen for predicting the direction of variation of the closing price of an exchange traded fund, DIA. The 'Diamonds' (AMEX ticker: DIA) is an exchange traded fund tracking the 30 corporations of the Dow Jones Industrial Average. The price of DIA, which is worth about 1/100th of the Dow Jones' value, faithfully tracks the fluctuations in the Industrial Average. We have decided to concentrate on this stock because DIA is a readily available instrument for trading on the Dow Jones Industrial Average, a canonical financial index.

Section 2 of this paper presents a description of the data set employed to predict the daily direction of variation of the DIA closing price. In Section 3 we describe the prediction model, which embodies a combination of GA and ANNs. In Section 4 we analyze the performance of the model on 63 test trading days on which the ANNs of the model have not been trained. Finally, in Section 5 we make some concluding remarks.

2. The input data

2.1. Data collection

Data selection and preprocessing constitute a crucial step in any modeling effort. The phases of data preparation can be broadly classified into three distinct areas: variable selection and collection, data inspection, and data preprocessing. Since our goal was to trade DIA frequently, we were able to narrow the list of choices to those variables relevant to the time frame. The data were obtained using a java-based query interface into a database of historical stock data available freely from Yahoo! (http://finance.yahoo.com).

A list of the variables employed in this study is shown in Table 1. The data were collected over the period from 11th November 2001 through 12th February 2003 (320 total trading days). All data were crosschecked to ensure accuracy. We have selected this data in order to provide a predictor model with a diversified database on which the ANNs can learn to extract input-output dependencies. We have decided to incorporate some of the major stock indices, as well as currency, bonds and gold prices, leaving the 'decision' of which indicators (or which non-linear combination of them) is predictive of the direction of variation of DIA to the model.

2.2. Data inspection

Even though the data were obtained from a reliable source, significant errors and missing data were present in the database. The validity of the input data was checked both visually and statistically. Missing data (holidays) were filled-in with the last trading day available. This filling-in has negligible consequences on the final performance of the system, since the data are pre-processed in the form of a normalized difference between the actual value and its

Table 1
The raw data loaded from Yahoo! database

Type	Name	Ticker (yahoo!)	Data structure
Stock	DIA (to be predicted)	dia	Open-High-Low-Close-Volume
Index	Nasdaq	^IXIC	Open-High-Low-Close-Volume
	S.Poor 500	^GSPC	Open-High-Low-Close-Volume
	Dow Jones Ind	^DJI	Open-High-Low-Close-Volume
	Dow Jones Trasp	^DJT	Open-High-Low-Close-Volume
	Dow Jones Util	^DJU	Open-High-Low-Close-Volume
	Dow Jones Compos	^DJA	Open-High-Low-Close-Volume
	Nikkei	^N225	Open-High-Low-Close
	Bovespa	^BVSP	Open-High-Low-Close
	Dax	^GDAX	Open-High-Low-Close
	Ftse 100	^FTSE	Open-High-Low-Close
Currency	Dollar/yen	^CJJ	Open-High-Low-Close
	Dollar/Swiss Frank	USDCHF = X	Open-High-Low-Close
Bonds	T-Bond 30 years	^TYX	Open-High-Low-Close
	T-Bond 10 years	^TNX	Open-High-Low-Close
	T-Bond 5 years	^FVX	Open-High-Low-Close
	T-Bond 13 years	^IRX	Open-High-Low-Close
	Eurobond	EUROD	Open-High-Low-Close-Volume
Commodities	Gold	^XAU	Open-High-Low-Close-Volume

The to-be-predicted stock, DIA, is loaded along with other historical values of indices, currencies, bonds and commodities. The sampling interval is from the 11th of November 2001 trough the 12th of February 2003 (320 total trading days). The raw input matrix consists of 85 indicators and 320 data points.

moving average. This pre-processing allows normalizing the indicator, therefore eliminating the bias given by its absolute magnitude in favor of a measure of its relative variation. This procedure allows using relatively long time series without the risk of overweighting the data with the higher absolute value.

2.3. Data pre-processing

In order to transform the data into a format acceptable by the prediction algorithms, derive a measure of the statistical fluctuations in the feature vectors used therein, and render the feature vectors independently of the absolute value assumed by the underlying indices from which the data are derived, two preprocessing operators were applied to the data. The renormalization operator (RNO) is used in some cases to extract daily changes in indicators that are independent of the absolute magniture of the indicator over time and to transform the fluctuations into percentages at others. The RNO operator is specified by

$$\frac{(x-y)}{y}100 \tag{3}$$

where x and y and take values specific to the indicator operated on. The Fluctuation Sensitive Function (FSF) is used when the absolute magnitude of an indicator should be retained for the prediction algorithms. This operator is simply specified by $x-y$.

After the raw data listed in Table 1 were loaded and crosschecked for accuracy, each field of this dataset, called dataset A, was independently normalized using the RNO operator. The data in the dataset A were further pre-processed and saved in a second dataset called dataset B, in order to reduce the learning load of the system and allowing the learning algorithms to concentrate the more predictive portions of the input (Table 2). This procedure spares the networks from extracting knowledge that can be provided directly in the input pattern, thereby allowing the system to extract higher-order relationships between these complex variables. The values indicated in Table 1 were preprocessed and re-arranged in the final input matrix, consisting of 64 data types and 320 data points. The pre-processing performed on the raw data is described in Tables 2 and 3. In general, the to-be-predicted stock data was extensively preprocessed, adopting a percentage measure of the difference between the indicator (Open, High, Low, Close, Volume) and its moving average (10 days), among other measurements of variation like Rate of Change (ROC), percentage difference between Open and Close, etc. Some widely used Technical Analysis indicators were also employed, namely Moving Average Convergence/Divergence (MACD), Relative Strength Index (RSI), Chainkin Volatility and MOMENTUM (Table 2). For the other indicators a measure of the variation given by Open, High, Low, and Close indicators and a Moving Average was used. More details on the nature of the pre-processing are reported in Tables 2 and 3. This new input matrix was further independently normalized for each field, resulting in each column assuming a value between 0 and 1. At the end of pre-processing, the absolute magnitude of all data was discarded, allowing the usage of an arbitrarily long dataset although the absolute values of the indicators part of the dataset might substantially vary over time. The two datasets A and B differ by the extensive preprocessing that characterizes the database B. Before being fed to the component networks of the prediction model, an additional preprocessing step called complement coding was applied. This is a normalization procedure in which a given input vector $\boldsymbol{x}$ is coded along with its complement $x^{\mathrm{c}} = (1-x)$. As a result, the input vector and the number of input nodes double in size, allowing automatic normalization of the input vector and, more importantly, an improvement of the pattern classification capabilities of the network (Carpenter, Grossberg, & Rosen, 1991). Complement coding allows a higher-order hidden node to be associated to both the presence and the absence of a given feature.

Finally, an additional preprocessing stage consists in the application of Principal Component Analysis (PCA). The input vector can therefore be substituted by a vector with the nth Principal Component of the original one.

Which dataset the network will be trained on, as well as the application of complement coding and PCA, are determined by the chromosome (further details in Section 3).

3. The model

The GA employed in this study is used to estimate appropriate parameters for m mixtures of networks (MON) used to predict the closing price of a security. The chromosome used by this GA has 11 'genes', which directly define the important parameters of the network, as well as the type of network and the input data type (database A or B, Table 3 for details). Every ith MON is composed of j networks to be selected from one of two types: Recurrent Backpropagation (Elman, 1991) or RBF networks (Duda et al., 2000). Elman networks are particularly suited for data when the temporal order of the data plays a crucial role (Elman, 1991), as it does in financial time series. RBF networks provide a highly compact representation for the underlying data distribution in cases where the bases accurately reflect the distribution. Each jth network is initialized by the chromosome and it is trained on the section of data defined, again, by the chromosome (Fig. 1). The chromosome defines network type, the architecture, the training set, and the most important parameters of the network as shown in Table 3.

Every network belonging to the ith MON is then tested on a blind data set, and single network responses are then conveyed into a voting procedure. Here a majority voting scheme chooses the prediction of the ith MON over the blind testing data, and assesses the fitness of the ith MON.

Table 2
This table summarizes the preprocessing stages applied to the raw data

Security	Operators	Equation
DIA	% ROC Close	$[(\text{Close}_t - \text{Close}_{t-1})/\text{Close}_{t-1}]100$
DIA	% Diff. Open-Close	$[(\text{Open} - \text{Close})/\text{Open}]100$
DIA	% Diff. High-Low	$[(\text{High} - \text{Low})/\text{Low}]100$
DIA	% Diff. Open and Mob. Avg. 10 dd	$[(\text{Open} - \text{Mavg10 dd})/\text{Mavg10 dd}]100$
DIA	% Diff. High and Mob. Avg. 10 dd	$[(\text{High} - \text{Mavg10 dd})/\text{Mavg10 dd}]100$
DIA	% Diff. Low and Mob. Avg. 10 dd	$[(\text{Low} - \text{Mavg10 dd})/\text{Mavg10 dd}]100$
DIA	% Diff. Close and Mob. Avg. 10 dd	$[(\text{Close} - \text{Mavg10 dd})/\text{Mavg10 dd}]100$
DIA	% Diff. Volume and Mob. Avg. 10 dd	$[(\text{Volume} - \text{Mavg10 dd})/\text{Mavg10 dd}]100$
DIA	Chaikin Volatility	H − L Average = Exponential moving average of(High − Low) $\left(\frac{\text{H} - \text{L Average}) - (\text{H} - \text{L Average n} - \text{periods ago}}{\text{H} - \text{L Average n} - \text{periods ago}}\right)100$
DIA	MACD	$\text{MACD}_t = \text{EMavg}_1 - \text{EMavg}_2$
DIA	MOMENTUM	$\text{Momentum} = \text{Close}_t - \text{Close}_{t-n}$
DIA	RSI	RS = (Avg. price change on up days − Avg. Price change on down days) RSI = 100 − (100/1 + RS)
Nasdaq	% Diff. Open and Mob. Avg. 10 dd	$[(\text{Open} - \text{Mavg10 dd})/\text{Mavg10 dd}]100$
S.Poor 500	% Diff. High and Mob. Avg. 10 dd	$[(\text{High} - \text{Mavg10 dd})/\text{Mavg10 dd}]100$
Dow Jones Ind	% Diff. Low and Mob. Avg. 10 dd	$[(\text{Low} - \text{Mavg10 dd})/\text{Mavg10 dd}]100$
Dow Jones Trasp	% Diff. Close and Mob. Avg. 10 dd	$[(\text{Close} - \text{Mavg10 dd})/\text{Mavg10 dd}]100$
Dow Jones Util		
Dow Jones Comp		
Nikkei Bovespa		
Dax		
Ftse 100		
Dollar/yen		
Dollar/Swiss Frank		
T-Bond 30 years		
T-Bond 10 years		
T-Bond 5 years		
T-Bond 13 weeks		
Eurobond	% Diff. Open and Mob. Avg. 10 dd %	$[(\text{Open} - \text{Mavg10 dd})/\text{Mavg10 dd}]100$
Gold	Diff. High and Mob. Avg. 10 dd	$[(\text{High} - \text{Mavg10 dd})/\text{Mavg10 dd}] \times 100$
	% Diff. Low and Mob. Avg. 10 dd	$[(\text{Low} - \text{Mavg10 dd})/\text{Mavg10 dd}] \times 100$
	% Diff. Close and Mob. Avg. 10 dd	$[(\text{Close} - \text{Mavg10 dd})/\text{Mavg10 dd}] \times 100$
	% Diff. Volume and Mob. Avg. 10 dd	$[(\text{Volume} - \text{Mavg10 dd})/\text{Mavg10 dd}] \times 100$
DJIA and T Bond 30 years	% Diff. between Normalized DJIA and Normalized T Bond	Norm. DJIA-Norm. Tbond

On the left column, the securities for which each operator is calculated are listed. The second column contains the name of the operator and the third column contains the equation by which the operator is calculated. ROC, rate of change; Diff., difference; MAvg, mobile average; EMAvg, exponential mobile average. Normalization: if $\min(x) > 0$, $x_{\text{norm}} = [(x\_\max - x\_\min)\cdot(x - \min(x))]/[(\max(x) - \min(x)]$, where $x\_\max$ and $x\_\min$ are the maximum and minimum value of the required mapping, respectively. If $\min(x) \leq 0, \min(x) = -|\max(x)|, x_{\text{norm}} = [(x\_\max - x\_\min)\cdot(x - \min(x))]/[(\max(x) - \min(x)]$.

The majority vote is simply given by:

$$\text{sign}\left[\sum_{j=1}^{n} y_j/n\right] \tag{4}$$

where y_j is the output of a component network, and n is the number of networks in a given MON.

In this study, the fitness of a MON is given by the percentage of correct responses over the blind data set. Note that a different measure of performance, namely the net return of the system over time, could have been employed. A correct response is defined as a match between the predicted and the actual direction of variation of the closing price of DIA in the following trading day.

It follows from this choice that the best MON in the population is not necessarily the one that has the lowest MSE on the training set, or the best jth network on the blind data set, but the one that expresses the highest number of correct voting predictions. This fitness function allows the dissociation of the performance on the training set for any single network to the actual generalization capabilities due to selection based on the collective voting measure, and on a blind data set that the component networks have not been trained on.

This choice follows the assumption that a single network cannot perform well on an extensive blind testing data set,

Table 3
The chromosome has 12 ‘genes’ coding different parameters of the networks

Gene #	Expressing these parameters	For this network type	Param. range (min–max)
0	Network type	RBF or ELMAN	0 or 1
1	Training epochs	E LMAN	20–1000
2	Learning rate	ELMAN	0.1–0.3
3	Type of input data (Database *A* or *B*)	RBF and ELMAN	0 or 1
4	Number of training data-upper bound	RBF and ELMAN	2
5	Number of training data-lower bound	RBF and ELMAN	n (where $n =$ max # of data)
6	Complement coding	RBF and ELMAN	0 or 1
7	# of PCA component	RBF and ELMAN	0–100
8	SSE Goal	RBF	0–10
9	Gaussian spread	RBF	0.1–10
10	Number of hidden layers-first layer	ELMAN	3–200
11	Number of hidden layers-second layer	ELMAN	3–200

The genes are differentially expressed depending on the type of network phenotype created by the chromosome [RBF or Recurrent Backprop (Elman)]. The parameter range for this simulation is shown on the right. For the genes at position 0, 3 and 6, a binary code (1 = on, 0 = off) was adopted.

but a family of networks trained on different time intervals could achieve better results. In this study a simple voting procedure was used. More sophisticated techniques could include super-ordinate networks that learn a nonlinear combination of the output of the component networks.

After fitness is calculated, a new set of MON is created using an even number of high-fitness populations of the past generation as a basis. In this simulation, the best four populations were selected for reproduction. The chromosomes specifying the networks of two of the winning populations are coupled with the corresponding chromosomes of another winning population, and crossover and mutation take place.

After crossover is performed between the n chromosomes of the component networks, the GA operator of mutation was applied (probability of mutation = 0.05).

The system was run for 100 generations, with 10 MON composed of 10 networks each (Elman and RBF). The four best MON were selected on the basis of their fitness, and were allowed to generate offspring (Figs. 2 and 3).

4. Results

In this simulation, 257 trading days of DIA are used for training and 63 trading days for testing. The final score is 73.4% correct up/down predictions over the blind data set,

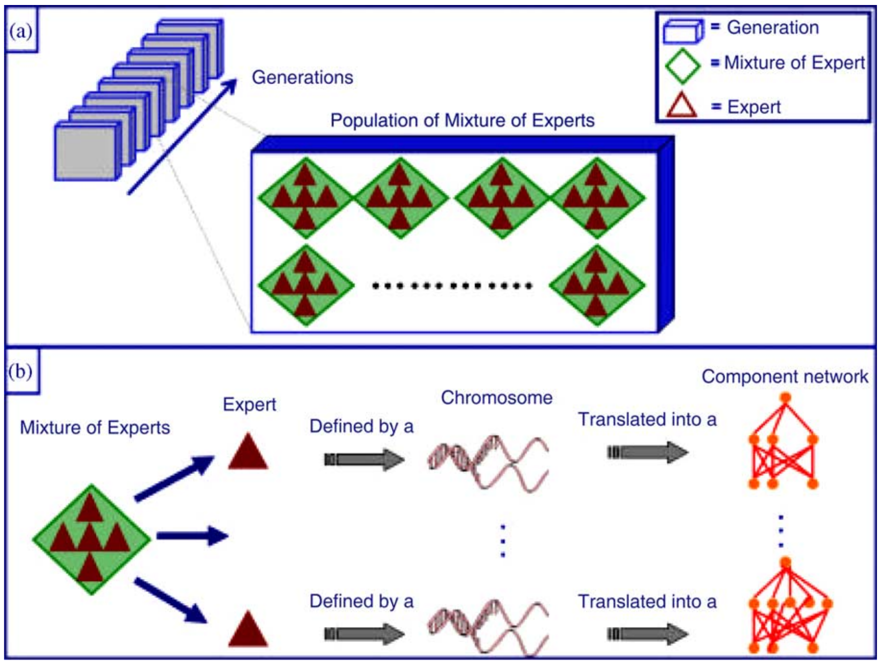


Fig. 1. (a) Every generation contains a population of MON which in turn contains n networks (Experts). (b) Each MON is composed of n networks (or experts), which are defined by a chromosome that specifies the architecture and parameters of the network.

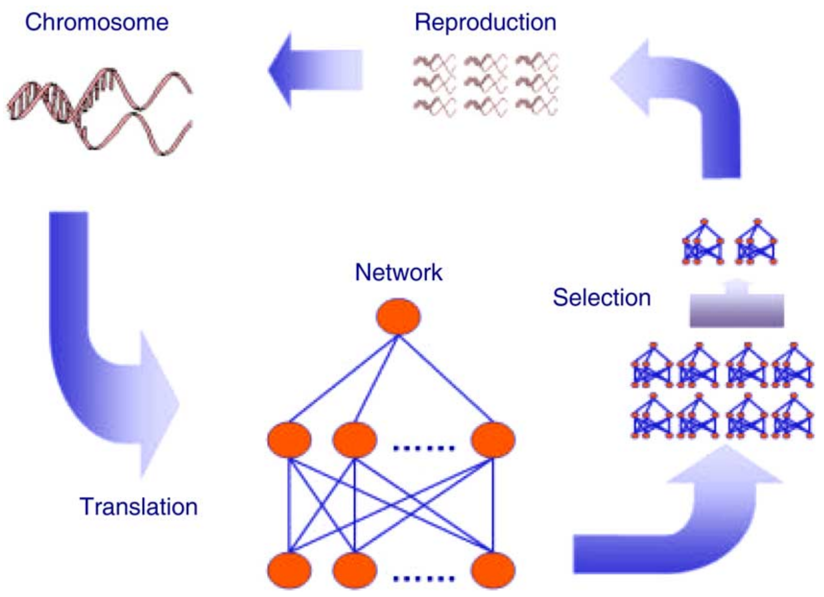
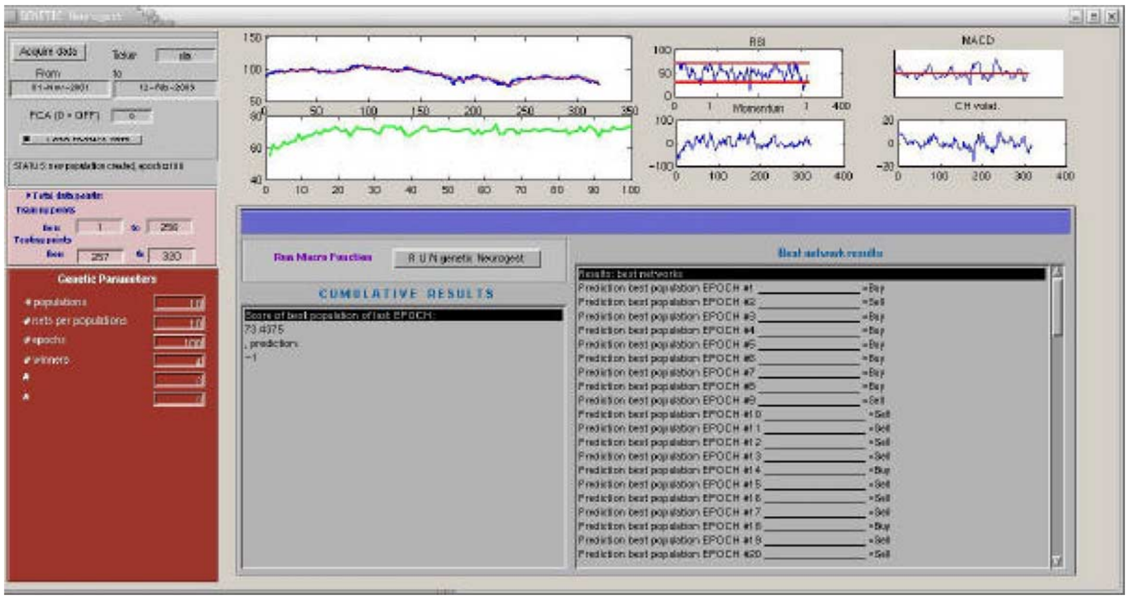


Fig. 2. The typical GA cycle. A chromosome defines the architecture of a network (in our study, a mixture of networks), which are then selected based on a measure of their fitness. The winning exemplars are then allowed to generate offspring, which is defined again trough a new chromosome.

whereas the best population scored 75.2% on the 63rd generation. A plot of the fitness of the best population across epochs is shown in Fig. 4.

In order to verify that the classifiers trained with the above data and tested using the blind datasets were not deviating significantly from the overall distributions of up and down days in the test dataset, we performed two tests. First, we performed a χ^2 test on the number of actual up/down days versus the number of up/down days predicted by the classifier. The hypothesis that the distribution of up/down predictions was the same as the observed distribution of up/down days in the test dataset could not be rejected ($\chi^2 = 0.0353$, critical value for confidence level of 0.05 for 1 DOF $= 3.84$). In order to analyze the bias of the classifier, we also performed a two-tailed pair wise t-test on the vectors signifying the up/down days (1 for up, -1 for down) for the test dataset and the classifier predictions. The hypothesis that the two vectors had different means could not be rejected either ($p = 0.5992$, two-tailed pair wise Student's t-test).

The above two tests strongly suggest that the classifiers trained using the methods described previously in this paper have a very low bias of prediction, if any.



```
STATUS: new population created, epoch [illegible]
Score of best population of last EPOCH:
73.4375
, prediction
-1
```

```
Results: best networks
Prediction best population EPOCH #1 ____________ =Buy
Prediction best population EPOCH #2 ____________ =Sell
Prediction best population EPOCH #3 ____________ =Buy
Prediction best population EPOCH #4 ____________ =Buy
Prediction best population EPOCH #5 ____________ =Buy
Prediction best population EPOCH #6 ____________ =Buy
Prediction best population EPOCH #7 ____________ =Buy
Prediction best population EPOCH #8 ____________ =Buy
Prediction best population EPOCH #9 ____________ =Sell
Prediction best population EPOCH #10 ____________ =Sell
Prediction best population EPOCH #11 ____________ =Sell
Prediction best population EPOCH #12 ____________ =Sell
Prediction best population EPOCH #13 ____________ =Sell
Prediction best population EPOCH #14 ____________ =Buy
Prediction best population EPOCH #15 ____________ =Sell
Prediction best population EPOCH #16 ____________ =Sell
Prediction best population EPOCH #17 ____________ =Sell
Prediction best population EPOCH #18 ____________ =Buy
Prediction best population EPOCH #19 ____________ =Sell
Prediction best population EPOCH #20 ____________ =Sell
```

Fig. 3. The GUI realized in MATLAB controls the main parameters of the GA, namely: training and testing intervals, number of populations, number of networks per population and number of winners per epoch.

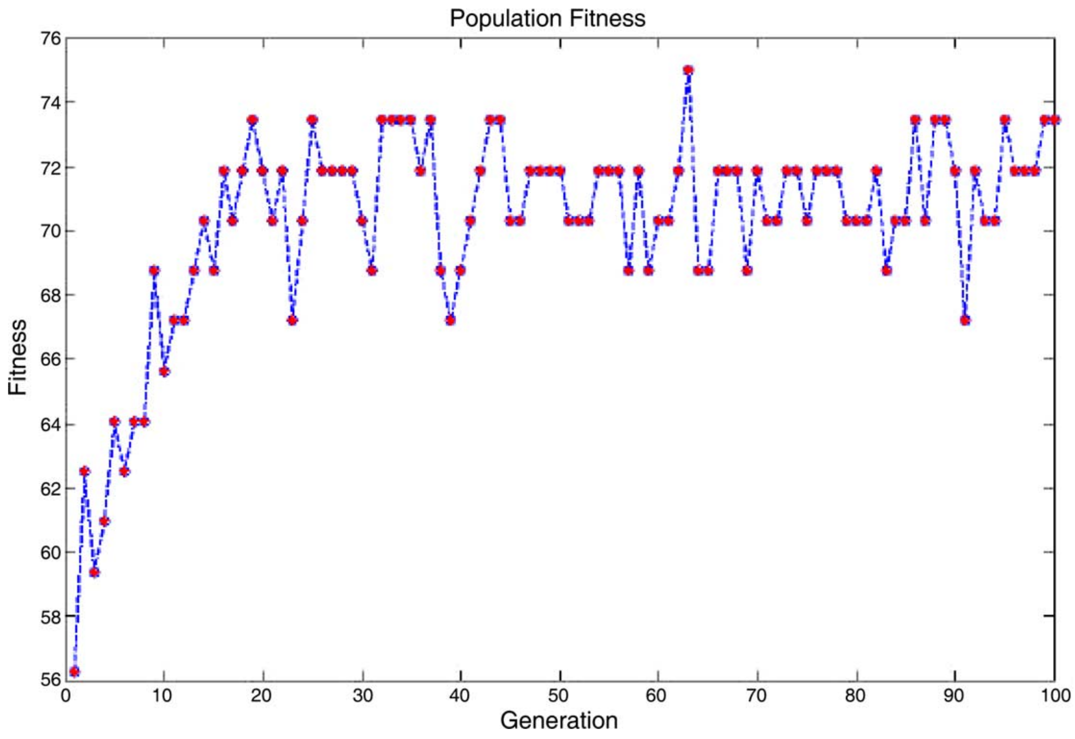


Fig. 4. The fitness of the best population across epochs shows a steep increase in the first 20 generations.

As can be seen in Fig. 4, the fitness of the best population of each epoch increases steeply in the first 20 epochs, reaches a peak on the 63rd generation, and oscillates between 66–74% in the following generations. These results are particularly good, and well above chance as shown by the χ^2 and t-test results.

5. Conclusions

The use of ANNs in financial applications has gained increasing popularity in the past decades. Nevertheless, a rigorous methodology on how to properly design a network or a system of networks that is able to successfully generalize from training to testing performance on a given time series is still lacking. This task is particularly difficult for large systems, especially when a combinatorial explosion of parameters is unavoidable due to the complexity and the magnitude of the system.

In this paper, we discuss a solution to the above problem employing a combination of ANNs and GAs that perform relatively well at predicting the closing price of a security. Whereas learning in ANNs can be considered homologous to ontogenetic learning, namely the adaptation of an organism to its environment throughout its lifespan, learning in GAs can be considered homologous to phylogenetic learning, which corresponds to the adaptive process running across generations through the selection of the best exemplars within a population. A system designed to model a complex problem such as financial forecasting should exploit the advantages and the differences of this two learning schemes. To conclude, this contribution shows that a combination of ANNs with GAs offer promise as a good technique for forecasting stochastic time series like those seen in stock market data.

References

Azoff, E. M. (1994). *Neural Network Time Series Forecasting of Financial Markets*. Chicester: John Wiley and Sons.

Below, R. K., McInnernay, J., & Schraudolph, N. (1990). *Evolving networks: using GAs with connectioninst learning. Technical Report CS90-174, Computer science and Engineering Department, University of California, San Diego*.

Carpenter, G. A., Grossberg, S., Rosen, D. B., & Fuzzy, A. R. T.:. (1991). *An adaptive resonance algorithm for rapid, stable classification of analog patterns. Proceedings of the International Joint Conference on Neural Networks (IJCNN-91), Piscataway, NJ: IEEE Service Center, II-411-416. Technical Report CAS/CNS-TR-91-006, Boston, MA: Boston University*.

Chang, E. J., & Lippmann, R. P. (1991). Using genetic algorithms to improve pattern classification performance. *Advances in Neural Information Processing*, *3*, 797–903.

Chu, C. H., & Widjaja, D. (1994). Neural network system for forecasting method selection. *Decision Support Systems*, *12*, 13–24.

Coleman, K. G., Graettinger, T. J., & Lawrence, W. F. (1991). Neural networks for bankruptcy prediction: The power to solve financial problems. *AI Review*, *5*, 48–50.

Duda, R. O., Hart, P. E., & Stork, D. E. (2000). *Pattern Classification* (2nd ed.). New York: Wiley-Interscience.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, *3*, 195–225.

Engle, R. F. (1982). Autoregressive conditional to select inputs with estimates of the variance of UK inflation. *Econometrica*, *50*, 987–1008.

Ginzburg, I., & Horn, D. (1994). Combined neural networks for time series analysis. *Advances in Neural Information Processing Systems Sci-Systems*, *6*, 224–231.

Goldberg, D. E. (1989). *Genetic algorithms in Search, Optimization and Machine learning*. Reading, MA: Addison Wesley.

Granger, C. W. J., & Anderson, A. P. (1978). *An Introduction to Bilinear Time Series Models*. Gottingen: Vandenhoeck and Ruprecht.

Harp, S. A., & Samad, T. (1991). Genetic synthesis of neural network architecture. In L. David (Ed.), *In Handbook of Genetic Algorithms*. New York: Van Nostrand Reinold.

Haykin, S. (1999). Neural networks: a comprehensive foundation. *IIIE*.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

Jordan, M., & Jacobs, R. (1994). Hsierarchical mixtures of experts and the EM algorithm. *Neural Computation Computation*, *6*(2), 181–214.

Jordan, M., & Xu, L. (1995). Convergence results for the EM approach to mixtures of experts architectures. *Neural Networks*, *8*(9), 1409–1431.

Kang, K., & Oh, J. (1997). Statistical mechanics of the mixture of experts. In M. Mozer (Ed.), (*vol. 9*) (pp. 183–189). *Advances in Neural Information Processing Systems*.

Kingdon, J. (1997). *Intelligent Systems and Financial Forecasting*. London: Springer Verlag.

Lapedes, A., Farber, R., (1987). Nonlinear signal processing using neural networks: prediction and system modeling. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, NM.

Lapedes, A., & Farber, R. (1988). How neural nets work. In D. Z. Anderson (Ed.), *Neural Information Processing Systems* (pp. 442–456). New York: American Institute of Physics.

Miller, G. F., Todd, P. M., & Hedge, S. U. (1989). Designing neural networks using genetic algorithms. *In Proceedings of the Third International Conference on Genetic Algorithms*, 379–384.

Montana, D. J., & Davis, L. (1989). Training feed-forward neural Networks using genetic algorithms. *In Proceedings of the International Joint Conference on Artificial Intelligence*, 746–767.

Odom, M. D., & Sharda, R. (1990). *A neural network model for bankruptcy prediction* (*vol. 2*). *In: Proceedings of the IEEE International Joint Conference on Neural Networks. San Diego, CA*, pp. 163–168.

Pelikan, E., de Groot, C., & Wurtz, D. (1992). Power consumption in West-Bohemia: Improved forecasts with decorrelating connectionist networks. *Neural Network World*, *2*(6), 701–712.

Refenes, A. N. (1995). *Neural Networks in the Capital Markets*. Chicester: Wiley.

Salchenkerger, L. M., Cinar, E. M., & Lash, N. A. (1992). Neural networks: A new tool for predicting thrift failures. *Decision Science*, *23*(4), 899–916.

Shaffer, J. D., Whitely, D., & Eshelman, L. J. (1990). On crossover os an evolutionary viable strategy. In R. K. Belew, & L. B. Booker (Eds.), *In Proceedings of the fourt International Conference on Genetic Algorithms* (pp. 61–68). San Mateo, CA: Morgan Kaufmann.

Tam, K. Y., & Kiang, M. Y. (1992). Managerial applications of neural networks: The case of bank failure predictions. *Management Science*, *38*(7), 926–947.

Trippi, R. R., & Turban, E. (1993). *Neural Networks in Finance and Investment: Using Artificial Intelligence to Improve Real-world Performance*. Chicago: Probus.

Tong, H., & Lim, K. S. (1980). Threshold autoregressive, limit cycles and cyclical data. *Journal of the Royal Statistical Society Series B*, *42*(3), 245–292.

Venkatesan, R., & Kumar, V. (2002). A genetic algorithms approach to growth phase forecasting of wireless subscribers. *International Journal of Forecasting*, *18*(2002), 625–646.

Waterhouse, S.R (2002). *Classification and Regression Using Mixtures of Experts*, Jesus College, Cambridge, and Department of Engineering, University of Cambridge, Dissertation submitted to the University of Cambridge for the degree of Doctor of Philosophy.

Weigend, A. S., Rumelhart, D. E., & Huberman, B. A. (1991). Generalization by weight-elimination with application to forecasting. *Advances in Neural Information Processing Systems*, *3*, 875–882.

Whiteley, D. (1989). The GENITOR algorithms and selection pressure: why rank-based allocation of reproductive trials is the best. In J. D. Shaffer (Ed.), *In Proceedings of the International Joint Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann.

Wilson, R., & Sharda, R. (1994). Bankruptcy prediction using neural networks. *Decision Support Systems*, *11*, 545–557.

Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, *14*, 35–62.