Ensemble Models in Forecasting Financial Markets

by Andreas Karathanasopoulos*, Mitra Sovan, Chia Chun Lo, Adam Zaremba and Mohammed Osman

***Corresponding author: Andreas Karathanasopoulos: Professor in Finance, University of Dubai, Dubai Business School, Academic City, P.O. Box: 14143, Dubai, UAE, e-mail: akarathanasopoulos@ud.ac.ae.**

Mitra Sovan: Associate Professor in Mathematical Finance, Liverpool University

Chia-Chun Lo: Associate Professor in Finance, Prince Mohammad Bin Salman College (MBSC)

Adam Zaremba: Associate Professor in Finance, University of Dubai & Poznan University of Economics and Business.

Mohammed Osman: Associate Professor in Economics and Dean of Business School, University of Dubai.

Abstract: In the current paper we study an evolutionary framework for the optimization of various types of Neural Networks structures and parameters. Three different evolutionary algorithms, named as Genetic Algorithms (GA) Differential Evolution (DE) and Particle Swarm Optimizer (PSO) are developed to optimize the structure and the parameters of three different types of Neural Networks: a Multilayer Perceptron (MLP) Recurrent Neural Network (RNN) and Radial Basis Function Neural Network (RBF). The motivation of this project is to present novel methodologies for the task of forecasting and trading financial indices. More specifically, the trading and statistical performance of all models is investigated in a forecast simulation of the SPY and the QQQ exchange-traded funds (ETFs) time series over the period January 2006 to December 2015 using the last 3 years as out-of-sample testing. As it turns out, the RBF-PSO, RBF- DE and RBF-GA ensemble methodologies do remarkably well and outperform all the other models.

Keywords: Financial Forecasting, Transaction costs, Multi-Layer Perceptron, Recurrent Neural Network, Radial Basis Function, Partial Swarm Optimizer, Genetic Algorithms, Differential Evolution.

1 Forecasting with ensemble models.

2 Forecasting Optimizers

3 Statistical and Empirical Perfomance

1. Introduction

Artificial neural networks have been popularly used for time series predictions since the 1980s. As Crone and Nikolopoulos (2007) mentioned in their recent research, till 2007 there have been around 5,000 publications in forecasting with neural networks applications. The most popular and well-known neural networks can be listed in 3 categories: multi-layer perceptron (MLP), recurrent neural networks (RNN) and radial basis function networks (RBF). The most popular learning algorithm that has been used for the training of the neural networks as Rumelhart et al. (1986) mentioned is the “back-propagation error” algorithm (BP) which can be described as "a milestone” in the field of forecasting with neural networks. However, there are still problems with neural networks when they are applied to real-world data such as 1) selection of the best inputs, 2) optimization of the weights of the neural network, 3) selection of the best fitness functions and 4) optimization in maximizing the forecasted output. In terms of overcoming the previous limitations, we are introducing novel ensemble neural network architectures of Particle Swarm, Genetic Algorithms, and Differential Evolution. More specifically, we propose fully adaptive architectures that decrease the number of parameters that practitioners need to set while on the other hand, this increases the forecasting ability of the neural network. In our paper, our proposed methodology comes in nine different versions where the three neural networks are combined with the three optimizers separately each time. Therefore we have an MLP combined with DE, GA and PSO optimizers secondly an RNN combined with DE, GA, and PSO optimizers and finally, an RBF combined with DE, GA and PSO optimizers in a forecasting and trading simulation of the SPY and the QQQ exchange-traded funds (ETFs) time series.

The rest of the paper is organized as follows: Section 2 presents the literature review which is focused on forecasting methodologies and, in particular, on ensemble neural networks. Section 3 describes the dataset used for the experiments and the descriptive statistics. Section 4 describes the models in this paper. Section 5 is the penultimate chapter which presents the empirical results and an overview of the benchmark models and in the final section, we present the conclusion.

2. Literature review

In the last decade, significant research on neural networks has been conducted, showing their superiority against other linear methodologies. Zhang and Berardi (2001), Hansen and Nelson (2003), Dunis et al. (2011), (2014) Karathanasopoulos et al. (2015a) (2016), Sermpinis et al (2012) and (2013), have published a number of papers, forecasting different stock markets, stating that multilayer perceptrons, higher order, and radial basis function neural networks outperform linear models such as the autoregressive moving average model, a naïve strategy, and a random walk model. Further, a new technique of combining linear with nonlinear or nonlinear with nonlinear models brought stronger performance in the area of forecasting. Nag and Mitra (2005) used a hybrid artificial intelligence method, based on a neural network and

genetic algorithm for modelling daily foreign exchange rates Sermpinis et al (2013) and Karathanasopoulos et al (2015b) have created ensemble models combining support vector machines with genetic algorithms in terms of forecasting the Greek stock market.

Being more specific in this paper among the various neural network techniques, we will use the Radial Basis Function (RBF) Neural Networks, Multilayer Perceptron (MLP) Neural Networks and Recurrent Neural Networks (RNN). The novelty of this project is the combination of all the previous neural networks with Particle Swarm Optimizers (PSO), Genetic Algorithms (GA) and Differential Evolution (DE). Kennedy and Eberhart (1995) mention that these types of combined algorithms optimize the neural network parameters, structure and training procedure. Because there is quite poor research comparing only ensemble models, we took the opportunity to test them and compare them for the task of forecasting two exchange-traded funds (ETFs). Karathanasopoulos et al (2010) and Irani and Nasimi, (2011) were the first to use the genetic algorithms combined with neural networks for stock indices predictions. Moreover, Zhang et al. (2007) and Sermpinis et al. (2013) combined the particle swarm optimizer with radial basis functions in the task of forecasting exchange currencies. Differential evolution optimizers (DE) have been used by Onwubolu and Davendra, (2006), Qu et al., (2012) and Wang et al., (2012) to select the initial connection weights and thresholds of neural networks. Vesterstrom and Thomsen (2004) mentioned that the differential evolution algorithm (DE) performs better than other popular intelligent algorithms, such as GA and PSO, based on 34 widely used benchmark functions. Compared with popular intelligent algorithms, the DE has less complex genetic operations because of its simple mutation operation and a one-to-one competition survival strategy. Furthermore, Wang et al. (2012) and Cai and Wang (2013) noticed that DE can also use individual local information and population to search for the optimal solution. However, even if DE is an easy to implement algorithm only a few researchers have used the DE to select suitable initial connection weights and thresholds in time series forecasting.

3. Data and Methods

3.1. Dataset

The study is based on exchange-traded funds, abbreviated ETFs. These investment vehicles are designed to replicate major stock indices, commodities, or other financial series, providing investors and easy access to various asset classes. Also, they offer to investors the opportunity to trade at very low transaction costs. The advantages of ETFs over “conventional trading” are well documented by Dolvin (2010) and Marshall *at al.* (2013).

In this research, nine ensemble adaptive neural networks are applied to the task of forecasting and trading two popular ETFs: SPDR S&P 500 ETF Trust (SPY) and the Invesco QQQ Trust.

SPY is the oldest and best-recognized ETF with one of the largest assets under management and the highest trading volume. It tracks the well-established S&P500 index representing the large-cap universe of US equities. It offers investors an easily-attainable exposure to a diversified portfolio of US stocks.

QQQ is also one of the most actively traded and best established ETFs. In contrary to SPY, it focuses solely on nonfinancial companies listed on NASDAQ. QQQ provides investors and exposure to – predominantly – US large-cap high-tech companies.

Both funds are listed in US dollars. We collect the source data from XXX. In particular, all ensemble models under the study forecast the one day ahead logarithmic returns of the two ETFs in two separated forecasting exercises. The relevant datasets are presented in Table 1.

Name of Period	Number of Trading Days	Beginning	End
Full Sample	3250	01/01/2005	31/12/2017
Training Period	2000	01/01/2005	02/01/2013
Validation Period	1250	03/01/2013	31/12/2017

Table 1. Total dataset

3.2. Methodology

In this study, three different evolutionary optimization techniques (GA, PSO, and DE) are applied to the task of optimizing the architecture of three neural networks (MLP, RNN, and RBF). More specifically, the three techniques optimize the number of hidden nodes and the center of the Gaussian Function for the RBF. The resultant nine hybrid neural network models are applied for the task of forecasting and trading the SPY and the QQQ ETF. Further to that in terms of feeding our ensemble neural networks, we are using an input selection based on the Model Confidence Set (MCS) test. An overview of the methodologies applied in this study is provided below.

3.2.1 MCS

The MCS test is a procedure that constructs a confidence interval which will contain the best model with a given level of confidence. It is a random data-dependent set of best forecasting models. More informative data can lead to only one best model while less-informative datasets will lead to multiple models. The MCS test is based on an equivalence test and an elimination rule. For a full description see Hansen et. al. (2010, 2011). In this study, the MCS procedure is applied to the task of selecting the NN inputs. More specifically it is applied in the in-sample test setting to the first ten lags of the ETF series presented in Table 2 below. The MCS confidence level is set to 95% while the criterion is the Mean Squared Error (MSE). This pool of ETFs consists of the two ETFs under study and 16 more ETFs that replicate indices, commodities, exchange rates and financial instruments possibly relevant to the series under study.

SPY (S&P 500)	AGG (US Bond)
QQQ (Nasdaq 100)	BND (Total Bond Market)
GDX (Gold)	VCSH (Short Term Corporate Bond)
EEM (Emerging Markets)	SHY (1-3 year US Bond)
IWM (Russell 2000)	SLV (Silver)
USO (Oil fund)	DBC (commodity Index)
EFA (EAFE)	SGOL (Swiss Gold Shares)
FXE (Euro/Dollar)	YCS (Yen/Dollar)
FXB (Pound/Dollar)	UUP (USDX futures index)

Table 2. MCS dataset

Note: The values in the table represent the relevant ETF symbol while in the parenthesis are the assets that the ETFs are replicating.

The MCS realizations (see Table 3 below) constitute the NNs inputs.

SPY	In-Sample Period	NN Inputs
	3/1/2013-31/12/2017	SPY(1), SPY (2), SPY (5), SPY(7), VTI(1), GDX(5), USO (3)

	3/1/2013-31/12/2017	SPY(1), SPY(2), SPY(3) SPY(8), GDX(3)
QQQ	3/1/2011-31/12/2017	QQQ(1), QQQ(3),QQQ(5), QQQ(6), EEM (3),GDX(5), USO (5)
	3/1/2013-31/12/2017	QQQ(1),QQQ(2),QQQ(5), GDX(3)

Table 3. MCS dataset

Note: The values in the table represent the relevant ETF symbols and the values to the parenthesis the lag.

We note from Table 3 that the lags of the forecasted series contain the relevant MCS realizations. In other words, the best forecasters for the SPY and the QQQ series are their own respective lags.

3.2.2 Neural Networks

Neural Networks are nonlinear models inspired by the work and function of biological neurons. They are used to approximate solutions in problems where the underlying function is unknown and nonlinear. The 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 intermediate 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 values 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.

Because of the stochastic nature of the proposed methodology, a simple run is not enough to measure its performance. This is the reason why 100 runs were executed and the average results are retained for evaluation.

MLP

The most popular and simple NN is the Multi-Layer Perceptron (MLP). A standard MLP neural network has three layers and this setting is adopted in the present paper. In general, its design follows the structure described in section 4.2. 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 initialized weights and proceeds by

applying a learning algorithm called backpropagation of errors[1] (Shapiro, 2000). The learning algorithm simply tries to find those weights which minimize an error function. Since networks with a sufficient number of hidden nodes are able to learn the training data (as well as their outliers and their noise), it is crucial to stop the training procedure at the right time to prevent overfitting (this is called 'early stopping'). This is usually achieved in the literature by dividing the in-sample dataset into two subsets respectively called the training and test sets used for simulating the data currently available to fit and tune the model. The network parameters are then estimated by fitting the training data using the above mentioned iterative procedure (backpropagation of errors). The iteration length is optimized by maximizing a fitness function in the test dataset. Finally, the predictive value of the model is evaluated by applying it to the validation dataset (out-of-sample dataset).

RNN

Our next model is the recurrent neural network. While a complete explanation of RNN models is beyond the scope of this paper, 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, 1991). A recurrent network uses feedback from one or more of its units as input in choosing the next output. This means that values generated by units at time step t -1, y(t-1), are part of the inputs x(t) used in the selecting the next set of values y(t) A simple recurrent network has activation feedback, which embodies short-term memory. The advantages of using recurrent networks over feedforward networks, for modeling 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", 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.

RBF

A radial basis function neural network (RBF) is a feedforward neural network where hidden units do not implement an activation function, but a radial basis function. In other words, they have the same structure as MLPs but the hidden nodes are based on the Gaussian function (instead of the sigmoid):

$$\phi^{[i]}(x_t) = e^{-\frac{\|x_t - C_i\|^2}{2\sigma_i^2}} \qquad (1)$$

where C_i is a vector indicating the center of the Gaussian function and σ_i is a value indicating its width. Compared with MLPs and RNNs the number of the weights between the input and the hidden layer is fixed to 1. Moreover, only the weights between the hidden layer and the output layer are adjusted during the training. This leads to a substantially decreased computational time. RBF approximates a desired function by superposition of nonorthogonal,

[1] 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)).

radially symmetric functions. They have been proposed by Broomhead and Lowe (1988) as an approach to improve the accuracy of artificial neural networks while decreasing the training time complexity.

3.2.3 Evolutionary Optimization

All the 3 neural networks described in the previous section have strong capabilities for processing nonlinear problems. However, substantial time is required for calculation, and the results obtained can easily fall into local optima. To improve these deficiencies we are combining the main neural network with three optimizers such as a Genetic algorithm, Particle swarm optimizer, and a differential evolutionary algorithm. In the next sections, there is a brief description of the three optimization techniques under study. For a complete discussion of GA, see Holland (1995), for PSO, see Eberhart and Kennedy (1995)(1997) and for DE, see Das and Suganthan (2011). The fitness function for the three techniques is the mean squared error.

GA

A genetic algorithm is an optimization algorithm inspired by the principle of natural selection (Holland (1995)). They have provided promising results in problems where the search space is big, complicated and there is no mathematical analysis available. Initially, a population of candidate solutions is generated. These solutions (called chromosomes) are optimized through a number of generations and two genetic operators (crossover and mutation). The chromosomes are consisting of genes that are the optimizing parameters. In each generation, all chromosomes are evaluated through a fitness function. The more fit chromosomes are selected to survive. Then, a sub-sample of the survivors is selected to evolve through one-point crossover or mutation.

One-point crossover creates two offsprings from every two parents. The probability of an individual to be selected as a parent is called crossover probability. A crossover point c_x is then selected randomly. The two offsprings are made by both concatenating the genes that precede c_x in the first parent with those that follow (and include) c_x in the second parent. The two offsprings replace their parents in the population. Mutation places random values in randomly selected genes with a certain probability (mutation probability). The mutation operator helps to avoid local optima and exploring a larger surface of the search space. In this study, the crossover and mutation probabilities are set to 0.9 and 0.1, respectively. Crossover is used with the aim that new chromosomes will have good parts of old chromosomes and maybe the new chromosomes will be better. However, it is often a fine technique to leave some parts of a population to survive to the next generation. This is the reason a high but not equal to one crossover probability was used. The selection step is based on the Holland (1995) guidelines. The initial population is set to 30 while the termination criteria are that either the algorithm reaches the 200th generation or the average fitness across the current population is less than 5% away from the best fitness in the last five generations.

PSO

Particle swarm optimization (PSO) is a stochastic optimization technique introduced by Eberhart and Kennedy (1995), and (1997). It is based on the social behaviour of birds within a flock. In the first stage, a random population of individuals (birds) is generated. Each individual in the population is called a particle. All particles are evaluated through a fitness function and have an initial fitness value. Each particle flies through the problem space with a specific velocity that directs the flight. The aim of all particles is to find the optimal solution. This is achieved through a number of iterations. In every iteration, all particles are evaluated and updated based on two values. The first value is the best value the particle has achieved so far (ibest). The second value is the best value of any particle that exists within the generation (gbest). Based on these two values, the particle updates its position based on equations:

$$x(t) = x(t-1) + v(t) \quad (2)$$

where x(t) is the position of the particle and v(t) is its velocity at generation t. The velocity is updated through :

$$v(t) = wv(t-1) + c_1 r_1\big(ibest(t) - x(t-1)\big) + c_2 r_2(gbest(t) - x(t-1)) \quad (3)$$

where c_1 and c_2 are the acceleration coefficients, r_1 and r_2 are random numbers between 0 and 1 that are used to drive the stochastic nature of the algorithm and w is a positive-valued parameter showing the ability of each particle to maintain its own velocity. The coefficients c_1 and c_2 represent the learning rates for the individual ability and the social influence respectively. Following the relevant literature, we set these values to 2. The initial population of particles is set to 30. The algorithm is stopped when the maximum number of iterations (set to 200) has been reached or the algorithm seems to have converged to an optimal solution. More specifically, the population of particles is deemed converged when the average fitness across the current population is less than 5% away from the best fitness of the current population fitness for more than five consecutive iterations. In this case, 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.

DE

The differential evolution (DE) algorithm is a stochastic real parameter optimization algorithm (Das and Suganthan (2011)) Similarly to the GA it is an evolutionary algorithm based on a sequence of iterations (generations) where individuals are updated through selection, mutation and crossover operators. The mutation operator perturbs population individuals with the scaled differences of randomly selected and distinct population members. In DE the potential solutions are represented as strings of continuous variables comprising feature and parameter variables. Discrete values are generated by rounding the continuous values. At the start, a random population is generated. The algorithm for every individual chromosome (X_i) of the population, selects three other random distinct individuals (($X_{1,i}$, $X_{2,i}$, $X_{3,i}$). Then a donor vector is produced:

$$V_i = X_{1,i} + F *(X_{2,i} - X_{3,i}) \quad (4)$$

where F is the mutation scale factor. A binomial crossover operator is applied which combines every member of the population X_i with its corresponding donor vector V_i and produces a trial vector U_i:

$$U_i(j) = \begin{cases} V_i(j) & , \quad if\,(rand_{i,j}[0,1] \le Cr) \\ X_i(j) & , \quad otherwise \end{cases} \tag{5}$$

At the next stage, all trial vectors are evaluated. If the value of the fitness function is lower than the one of the population X_i then the trial vector takes its position in the population. Thus the population is only getting better through the iterations. The parameter F controls the size of the differentiation quantity which is going to be applied to a candidate solution from the mutation operator while Cr determines the number of genes that are expected to change in a population. In this study, we follow the guidelines of Qin *et. al.* (2009): F is selected from a uniform distribution with mean value 0.5 and standard deviation 0.3 and Cr is selected from the same distribution with a mean 0.5 and standard deviation 0.1. The size of the initial population is 30. Similarly to GA and PSO, the termination criterion is a combination of the maximum number of iterations (which is set to 200) and a convergence criterion. The convergence criterion is satisfied when the mean population fitness is less than 5% away from the best population fitness for more than five consecutive iterations

4. Empirical results

4.1 Statistical Performance

In terms of checking the statistical performance of our combined models, we have used the below statistical measures. In tables 4 and 5 the statistical performance in the out-of-sample period of all models is presented. We utilize three measures of forecasting accuracy: the Mean Absolute Error (MAE), the mean absolute percentage error (MAPE), and the Root Mean Squared Error (RMSE). These three metrics are measures of the difference between the predicted and actual ETF returns. MAE averages the vertical distance between the two variables, MAPE is based on the mean absolute percentage difference, and RMSE focuses on the root of the squared differences between individual forecast and actual realizations. All the three measures are defined in such way that the lower the output, the better the forecasting accuracy of the model is. For further discussion on the alternative measures, see, e.g., Hyndman and Koehler (2006).

The Diebold-Mariano (1995) (DM) statistic for predictive accuracy statistic tests the null hypothesis of equal predictive accuracy between two models. In this exercise, the second model is always a buy and hold strategy. A statistically significant difference between the predictive accuracies indicates that the forecasts of the first model (NN-evolutionary optimizer) are better than a buy and hold strategy. The Pesaran-Timmermann (1992) (PT) test examines whether the directional movements of the real and forecast values are moving close one to another. Furthermore, it checks how well rises and falls in the forecasted value follow the actual rises

and falls of the time series. The null hypothesis is such that the model under study has 'no predictive power' when forecasting the ETF return series.

Forecast	MLP-PSO	MLP-DE	MLP-GA	RNN-PSO	RNN-DE	RNN-GA	RBF-PSO	RBF-DE	RBF-GA
MAE	0.0300	0.0290	0.0280	0.0221	0.0211	0.0190	0.0150	0.0130	0.0160
MAPE	200.17%	198.12%	196.60%	189.33%	183.22%	181.22%	160.91%	150.19%	162.11%
RMSE	0.0354	0.0327	0.0287	0.0167	0.0205	0.0184	0.0153	0.0145	0.0162
PT-Statistic	10.22	10.76	10.81	11.13	11.44	11.81	14.12	14.80	14.34
DM	-3.45	-3.21	-2.94	-2.44	-2.31	-2.14	-1.67	-1.14	-1.45

Table *4: Statistical performance of SPY (out of sample period)*

Forecast	MLP-PSO	MLP-DE	MLP-GA	RNN-PSO	RNN-DE	RNN-GA	RBF-PSO	RBF-DE	RBF-GA
MAE	0.0382	0.0360	0.0321	0.0290	0.0287	0.0265	0.0231	0.0223	0.0254
MAPE	311.11%	310.13%	302.72%	291.77%	290.26%	270.33%	222.91%	210.01%	233.54%
RMSE	0.0455	0.0412	0.0401	0.0380	0.0350	0.0312	0.0240	0.0230	0.0260
PT-Statistic	12.55	12.67	12.90	13.22	13.40	14.30	15.00	15.39	14.89
DM	-5.02	-4.12	-4.01	-3.89	-3.45	-3.25	-2.89	-2.56	-2.91

Table 5: Statistical performance of QQQ (out of sample period)

From the table above we note that the RBF DE outperforms all benchmarks for all of the statistical measures. The RBF-PSO model presents the second best performance while the RBF-GA algorithm produces the third most statistically accurate forecasts. The PT statistics indicate that all non-linear models under study are able to forecast accurately the directional movements of the three return indices while the DM statistics confirm the statistical superiority of the RBF-DE.

4.2 Empirical Trading Results

In this section, we present the results of the proposed methodology applied to trade the S&P 500 and Nasdaq 100 exchange trade funds in the relevant out-of-sample period. The trading rule that we follow is simple. We go or stay long when the forecast return is above zero and go

or stay short when the forecast return is below zero. Transaction costs are estimated as 16 basis points or 0.16% per position for both ETFs (see, www.interactivebrokers.com)

	MLP-PSO	MLP-DE	MLP-GA	RNN-PSO	RNN-DE	RNN-GA	RBF-PSO	RBF-DE	RBF-GA
Annualized Return (including costs)	9.44%	10.00%	11.00%	12.03%	12.66%	13.42%	17.91%	18.03%	16.46%
Sharpe ratio	0.70	0.67	0.78	0.95	0.98	0.97	1,39	1,45	1,23

Table 6: Out of sample trading performance results for the SPY (including cost)

	MLP-PSO	MLP-DE	MLP-GA	RNN-PSO	RNN-DE	RNN-GA	RBF-PSO	RBF-DE	RBF-GA
Annualized Return (ixcluding costs)	13.33%	14.00%	14.22%	16.17%	16.55%	16.59%	20.01%	20.81%	19.01%
Sharpe ratio	0.92	0.95	0.96	0.99	1.01	1.00	1.22	1.33	1.11

Table 7: Out of sample trading performance results for the QQQ (including cost)

We can see that the RBF-DE performs significantly better than all the other benchmark methods in terms of Sharpe ratio and annualized return. The RNN combinations come second while the MLP models present the worst trading performance. It is noticeable that all models present a positive trading performance for both ETFs. In addition to this, the output of the empirical analysis proves the advantage of combining neural networks with optimizers by minimalizing the time of the calculation and at the same time by reducing the square error significantly.

5. Conclusion

In this study, three different evolutionary algorithms, namely a GA, a DE and a PSO are applied to the task of optimizing the structure and the parameters of three different types of Neural Networks: Multilayer perceptron (MLP) Recurrent Neural Network (RNN) and Radial Basis Function (RBF). The generated models are applied to the task of forecasting and trading the daily closing prices of SPY and QQQ ETF. These data series are representing the exchange traded funds of NASDAQ 100 and S&P 500 general American indices.

By converting all our input candidates from table 2 to autoregressive returns we start backtesting the ensemble structures. Finding the best data optimised inputs from the backtest

we start feeding and testing our ensemble models. All nine ensemble NN-evolutionary models have been managed to trade successfully the two ETFs. The RBF hybrid combination models present the best performance, the second performance its coming from the RNN combinations while the MLP architectures provide the third lowest performance. All the combinations show a remarkable performance proving that hybrid combinations are improving the final performance of the model and in continuation, they are making good profit returns. In the meantime, we have to add that running ensemble model we need less time than running simple NNs.

In conclusion, it's worth saying that these results demonstrate the utility of evolutionary algorithms in NNs parametrization. They also reveal the performance of hybrid NN and Heuristics models in financial trading. These results should go forward on convincing academic and practitioners to further experiment with NNs and evolutionary optimizers in the search for abnormal profits.

References

Broomhead, D., & Lowe, D. (1988). Multivariate functional interpolation and adaptive networks. *Complex Systems*, 2, 321-355.

Cai, Y., and Wang, J. (2013). Differential evolution with neighborhood and direction information for numerical optimization. *IEEE Transactions on Cybernetics*, 43 (6), 2202–2215.

Crone, S., & Nikolopoulos, K. (2007). Results of the NN3 neural network forecasting competition. *Proceedings of the 27th International Symposium on Forecasting Program*, 129.

Das, S., & Suganthan, P, N. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15 (1), 4-31.

Diebold, F, & Mariano, R. (1995). Comparing predictive accuracy. *Journal of Business and Economic Statistics*, 13, 253–263

Dolvin, S, D. (2010). S&P ETFs: Arbitrage opportunities and market forecasting. *Journal of Index Investing*, 1 (1), 107-116.

Dunis, C., Karathanassopoulos, A., and Laws, J., (2011) Modelling and trading the Greek stock market with mixed neural network models, Applied Financial Economics 21 (23), pp. 1793-1808

Dunis , C., Karathanasopoulos, A., Sermpinis, and G., Laws, J.(2014) Modelling and Trading the Greek Stock Market with Gene Expression and Genetic Programing Algorithms, Journal of forecasting, 33 (8), pp. 596-610.

Elman, J. L. (1990). Finding structure in time. Cognitive Science, 14, 179–211.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7, 195–224.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. *Proceedings of the IEEE international conference on neural networks*, 4, 1942–1948).

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. *Proceedings of the conference on systems, man, and cybernetics*, 4104–4109.

Hansen, P., & Lunde, A. (2010). An oxtm software package for multiple comparisons. Available from: http://mit.econ.au.dk/vip_htm/alunde/mulcom/mulcom.htm.

Hansen, P., Lunde, A., & Nason, J. (2011). The model confidence set. *Econometrica*, 79, 453–497.

Hansen, J, V., & Nelson, D. (2003). Forecasting and recombining time-series components by using neural networks. *Journal of the Operational Research Society*, 54 (3), 307–317.

Holland, J. H. (1995). *Hidden order: How adaptation builds complexity?* Addison-Wesley, Redwood City, CA.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22 (4), 679–688.

Irani, R., & Nasimi, R. (2011). Evolving neural network using real coded genetic algorithm for permeability estimation of the reservoir. *Expert Systems with Applications*, 38 (8), 9862–9866.

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

Karathanasopoulos, A., Theofilatos, K., Leloudas, P., and Likothanassis, S. (2010) Modelling the ASE20 Greek index using artificial neural networks combined with genetic algorithms. In Proceeding of the 20th International Conference on Artificial Neural Networks, pp.428-435.

Karathanasopoulos, A., Stasinakis, C., Sermpinis, G., and Theofilatos, K., (2015a) Modeling, forecasting and trading the EUR exchange rates with hybrid rolling genetic algorithms—Support vector regression forecast combinations, European Journal of operation research, 247 (3) pp. 831-846.

Karathanasopoulos, A., Sermpinis, G., Stasinakis, C., and Theofilatos, K., (2015b)Forecasting US Unemployment with Radial Basis Neural Networks, Kalman Filters and Support Vector Regressions, Computational economics, pp1-19.

Karathanasopoulos, A., Theofilatos, k., Sermpinis G., Dunis, C., and Mitra, S., (2016a) Stock market prediction using evolutionary support vector machines: an application to the ASE20 index, The European Journal of Finance 22 (12), pp. 1145-1163

Karathanasopoulos, A., (2016b) Modelling and trading the English stock market with novelty optimization techniques Economics and Business Letters 5 (2), pp.50-57

Marshall, B., Nhut N., & Visaltanachoti, N. (2013). ETF arbitrage: Intraday evidence. *Journal of Banking & Finance*, 37, 3486-3498.

Nag, A, K., & Mitra A. (2005). Multiple outlier detection in multivariate data using self-organizing maps. *Computational Statistics*, 20 (2), 245-264.

Onwubolu, G., & Davendra, D. (2006). Scheduling flow shops using differential evolution algorithm. *European Journal of Operational Research*, 171, 674–679.

Pesaran, M., & Timmermann, A. (1992) A simple nonparametric test of predictive performance. *Journal of Business and Economic Statistics*, 10, 461-465.

Qin, A, K., Huang, V, L., & Suganthan, P, N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 13, 398–417.

Qu, B, Y., Suganthan,P, N., & Liang, J. (2012). Differential evolution with neighborhood mutation for multimodal optimization. *IEEE Transactions on Evolutionary Computation*, 16 (5), 601–614.

Rumelhart, D, E., Hinton, G, E., & Williams, R, J. (1986). Learning representation by backpropagating errors. *Nature*, 232, 533–536.

Sermpinis, G., Karathanasopoulos, A., Laws, J., and Dunis, C., (2012) Forecasting and trading the EUR/USD exchange rate with Gene Expression and Psi Sigma Neural Networks , Expert systems with applications , 39 (10,) pp. 8865–887.

Sermpinis, G., Karathanasopoulos, A., Theofilatos, K., Georgopoulos, E., Dunis, C. (2013) "Forecasting Foreign Exchange Rates with Adaptive Neural Networks Using Radial-Basis Functions and Particle Swarm Optimization." European Journal of Operational Research, 225(3), pp. 528-540.

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

Tenti, P. (1996). Forecasting foreign exchange rates using recurrent neural networks. *Applied Artificial Intelligence*, 10 (6), 567-581.

Vesterstrom, J., &. Thomsen, R. (2004). A comparative study of differential evolution, particle swarm optimization and evolutionary algorithms on numerical benchmark problems. *Proceedings on Congress on Evolutionary Computation*, 2, 1980-1987.

Wang, L., He, J., & Zeng, Y, R. (2012). A differential evolution algorithm for joint replenishment problem using direct grouping and its application. *Expert Systems*, 29 (5), 429–441.

Zhang, G, P., & Berardi, V, L. (2001). Time series forecasting with neural network ensembles: an application for exchange rate prediction. *Journal of the Operational Research Society*, 52 (6), 652–664

Zhang, J, R., J. Zhang, J., Lok, T. M., & Lyu M, R. (2007). A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training. *Applied Mathematics and Computation*, 185 (2), 1026-1037.

.