NEURON OPTIMIZATION OF EVOLUTIONARY ARTIFICIAL NEURAL NETWORKS FOR STOCK PRICE INDEX PREDICTION

Asil ALKAYA
Adnan Menderes University / Faculty of Business Administration
Title: Asst. Prof. Dr.
E-mail: asil.alkaya@adu.edu.tr

─Abstract ─

This study presents an optimization procedure for the number of processing elements (neurons) of hidden layers to predict a stock price index using Evolutionary Artificial Neural Networks (EANN), in particular, for the Istanbul Stock Market price index (ISE) in order to contribute to the development of Intelligent Systems Methods for modeling several systems that are highly non-linear and uncertain.

The US dollars/Turkish Lira (US/TRY) exchange rate, Euro/Turkish Lira (EUR/TRY) exchange rate, ISE National 100 (XU100) index, world oil price, and gold price were used as for a period of approximately 10 years' daily data as inputs. Performance is benchmarked by mean squared error, normalized mean squared error; mean absolute error and the correlation coefficient. With the fixed neural network architecture and optimized parameters, evolutionary neural networks perform better performance values when the number of neurons used in hidden layers is optimized.

Key Words: *Evolutionary algorithms, artificial neural networks, prediction, stock price index.*
JEL Classification: C45, C53, C61

1. INTRODUCTION

Neural networks are a hill climbing search. As such, they are subject to the pitfalls of getting stuck on local features of the solution space. Neural networks use an error calculation to compute a gradient to direct the search; such as the backpropagation network. These methods require smooth, continuous activation functions in order to derive gradient information.

Evolutionary (genetic) algorithms do not perform direct calculation of gradients. Instead, they focus on blanketing the search space with potential solutions. This results in a far more global search which is much less likely to succumb to local features of the solution space. These advantages give evolutionary algorithms a much wider range of options; an evolutionary algorithm might use linear thresholds, splines, or product units where traditional neural networks might require a smooth sigmoid function. Further, computation of gradients in the more complex neural networks, such as recurrent networks, can be quite costly. EAs do not require these expensive calculations.

2. EVOLUTIONARY ARTIFICAL NEURAL NETWORKS

Traditional artificial neural networks based on backpropagation algorithms have some limitations. At first, the architecture of the artificial neural networks is fixed and a designer needs much knowledge to determine it. Also, error function of the learning algorithm must have a derivative. Finally, it frequently gets stuck in local optima because it is based on gradient-based search without stochastic property. While the underlying algorithms may be relatively simple; network parameters such as learning rate, momentum, initial weights, number of layers, and number of units per hidden layer play a large part in the ability of a particular network to solve a given problem.

Even when selected and implemented by an expert with knowledge both of neural networks and the problem domain, the process is often little better than trial and error. A better way to determine optimal parameter settings for a neural network is required. The goal of apply-

ing an evolutionary algorithm is to automate ad hoc process of neural network design.

The main steps for EANN based prediction is as follows:
i. Generate the initial population with multi-layer perceptrons with random weight values in a specified range and specified initial number of hidden layer sizes.
ii. Repeat for n generations:

- Evaluate the new MLP's (individuals): train them using the training set and obtain their fitness according to the number of correct classifications on the validation set and the network size (number of weights).
- Select the s best individuals in the population, according to their fitness, to mate using the genetic operators to obtain the new individuals.
- Replace the s worst individuals in the population by the new ones.

iii. Use the best individual found to obtain the testing error using the test set.

Figure 1. EANN framework for prediction

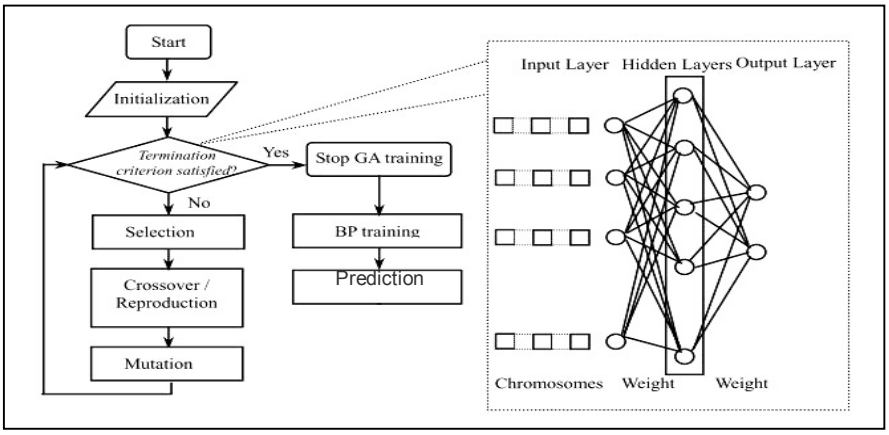
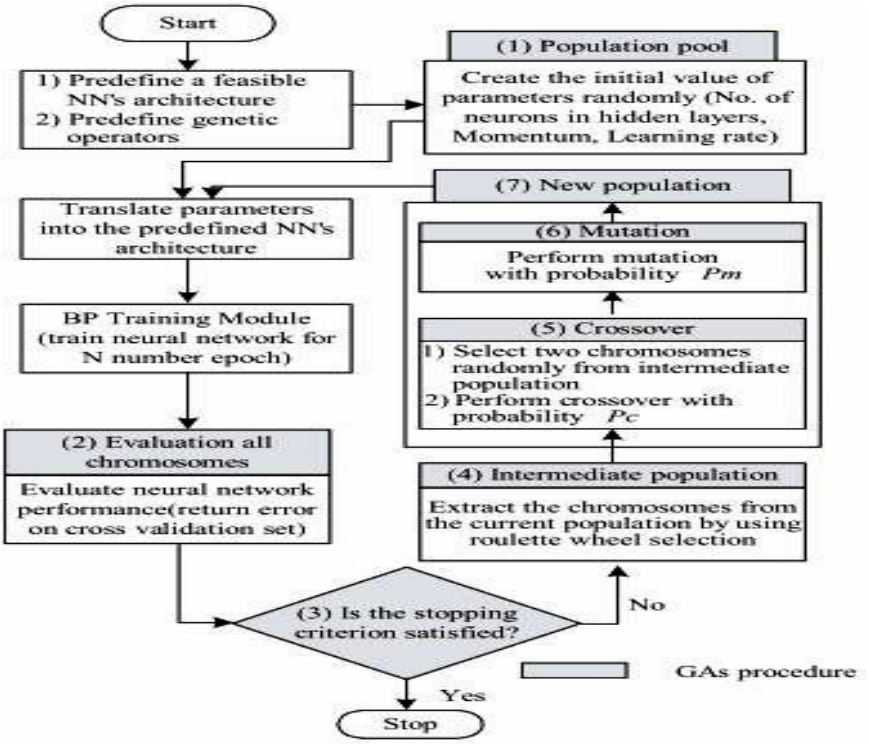


Figure 2. Neuron optimization procedure



The longer the chromosomes the more generations are required. In general, genetic algorithms are inherently slower than backpropagation. This could be expected due to their global search technique compared to the highly directed gradient descent learning of backpropagation. During the EANN prediction process; to identify the neural network parameters, the evolving mechanism is used such as mutation and crossover. Genetic algorithms are used to determine the number of neurons in the hidden layers, the momentum, and the learning rates for minimizing the time and effort required to find the optimal architecture and parameters of the back-propagation algorithm.

3. PARAMETERS FOR EANN BASED PREDICTION

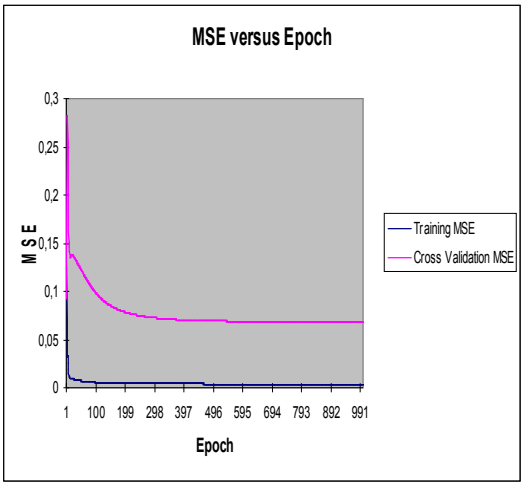
The US dollars/Turkish Lira (US/TRY) exchange rate, Euro/Turkish Lira (EUR/TRY) exchange rate, world oil price, and gold price were used as inputs for predicting Istanbul Stock Market price index. Starting from January 2, 2002 up to February 17, 2012; all the data collected on a daily basis gathering a total number of 10327 exemplars.

The number of hidden layer size is important up to the required level. One hidden layered EANN is not suitable for prediction and also increasing the number of hidden layer causes a big decrease in the training and cross validation subsets' mean accuracy values. For three hidden layered topology, overtraining occurs so that as a universal mapper, two hidden layered topology is the optimum.

The number of chromosomes in the evolutionary algorithm plays an important role on processing time. This parameter is chosen by trial and error. The minimum and average MSE values give opinion about the parameter performance. Because much longer chromosome type causes longer processing times, shorter possible chromosome length should be selected. When the population size was set down from 50 to 20 and the generation number to 1000, the run values were nearly the same. The processing time decreased down from 6,3 to 2,8 hours on average.

The selection of the percentage of the whole raw data to split into training, cross validation and test subset is important. For dataset, there are 2542 exemplars (four inputs as a set) in total. In this study, the splitting percentages are; 60% for training subset (1525 exemplars), 20% for cross validation subset (509 exemplars) and 20% for testing subset (508 exemplars).

Figure 3. Performance values of EANN for hidden layer 1=30, hidden layer 2=15



Best Networks	Training	Cross Validation
Epoch #	1000	1000
Minimum MSE	0,0379	0,068
Final MSE	0,0379	0,068

For higher values of crossover, crossover=0.9, heuristic mutation is much more efficient than the other mutation types taken into consideration.

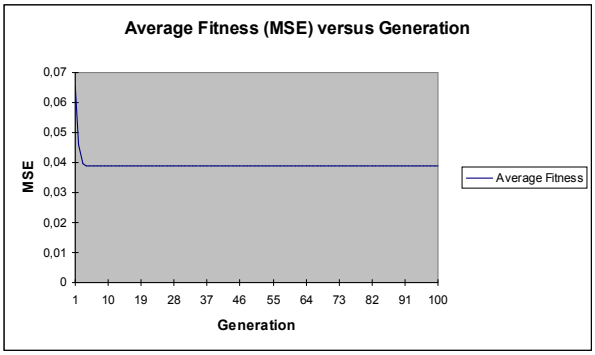
Best prediction for the test data is made for crossover=0,5 and mutation=0,3 that has the heuristic crossover type.

3.1 Neuron optimization for hidden layers

From experiences of the study, due to the termination value of $1x10^{-5}$, the optimum number of hidden layers is two and the optimum number of hidden neurons in the hidden layer 1 is 13 and the optimum number of hidden neurons in the hidden layer 2 is 3. At that point, these values for the parameters are found at fixed weight matrix and no genetic operators are implemented to the neural network design.

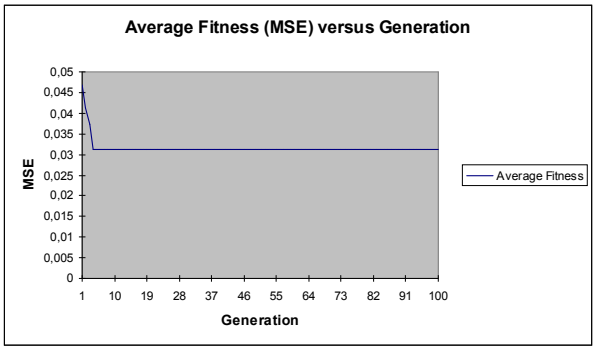
By embedding the evolutionary algorithm to the network weight space, new neural network architectures are gained in a population pool. New parameters extracted from the genetic operators are put in the proposed neural network and statistical performance values (minimum MSE and average MSE) are taken out. For each genetical scenario, the statistical results are gathered as a performance.

Figure 4. Performance values of EANN for hidden layer 1=13, hidden layer 2=15



Optimization Summary	Best Fitness	Average Fitness
Generation #	1	6
Minimum MSE	0,0387	0,0387
Final MSE	0,0387	0,0387

Figure 5. Performance values of EANN for hidden layer 1=13, hidden layer 2=3



Optimization Summary	Best Fitness	Average Fitness
Generation #	1	4
Minimum MSE	0,0313	0,0313
Final MSE	0,0313	0,0313

As comparing the performances of Figure 3 and Figure 4; by reducing the neurons of hidden layer 1 from 30 to 15, the final MSE values just got 2,11% worse and also reducing the neurons of hidden layer 2 from 15 to 3, the final MSE got 17,41% better for the fitness values of EANN.

Although, as the dimension of hidden layer decreases and fitness values get better, the predicted values got worse due to reduced connection of layers of EANN and the training ability got weak.

Figure 6. Prediction for neuron size=30 for hidden layer 1 and neuron size=15 for hidden layer 2.

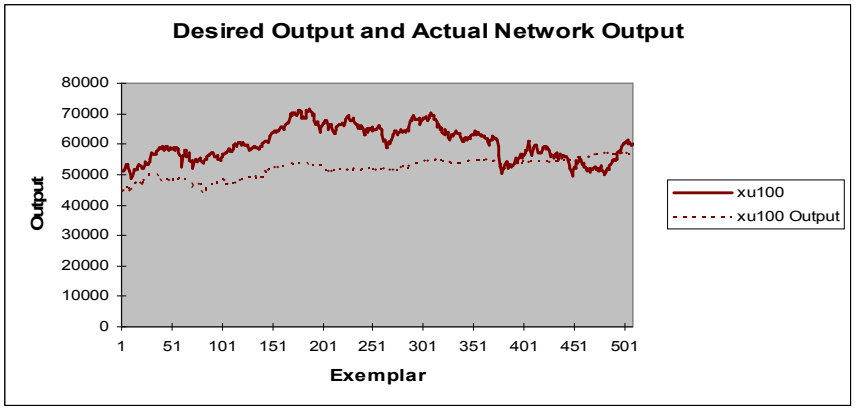
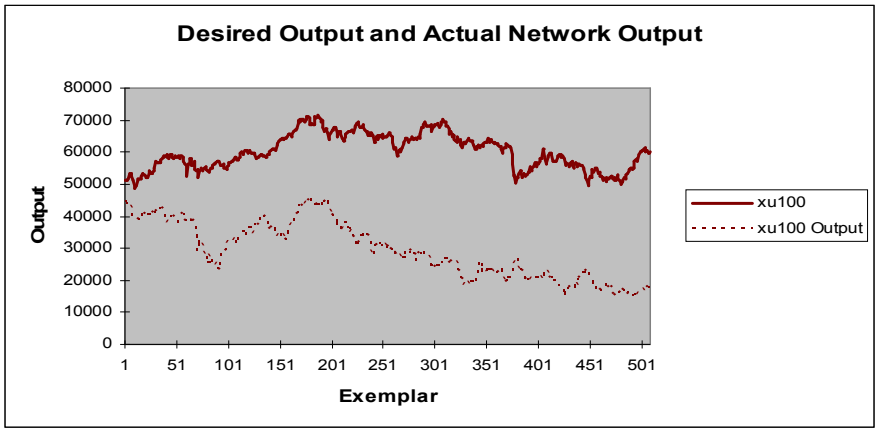


Figure 7. Prediction for neuron size=13 for hidden layer 1 and neuron size=3 for hidden layer 2.



MSE of prediction values of Figure 6 is 99613647,75 and MSE of prediction values of Figure 7 is 103259157,6 so that neuron optimized EANN has just 3,66% worse than the former EANN.

3. CONCLUSION

The number of hidden layer size is important up to the required level. But increasing the number of hidden layer more than needed causes a big decrease in the training and cross validation subsets' mean accuracy values. Therefore, two hidden layer is suitable for prediction. Optimized number of neurons for each hidden layer gives advantage of shortened processing time and more accurate performance values but as a trade-off; because the size of EANN gets smaller, training ability lessens and the level of accuracy of prediction gets downwards in a small percentage.

BIBLIOGRAPHY

Alkaya A. & Bayhan, G.M. (2009). The Classification of a Simulation Data of a Servo System via Evolutionary Artificial Neural Networks, *International Conference on Intelligent Computing Proceedings*, pp 48-54.

Ang J.H., Tan K.C. & Al-Mamun A. (2008). Training neural networks for classification using growth probability-based evolution, *Neurocomputing* ,71 3493–3508

Blum, A., (1992). *Neural networks in C++: an object-oriented framework for building connectionist systems*, John Wiley & Sons, Inc., pp. 86-103

Castillo-Valdivieso, P. A., Merelo J. J., & Prieto A. (2002). Statistical Analysis of the Parameters of a Neuro-Genetic Algorithm, *IEEE Transactıons On Neural Networks*, Vol. 13, No. 6.

Fine,T.L. (1999). *Feedforward Neural Network Methodology*, Springer, New York, pp. 129-194

Freitas A. (2002). A Survey of Evolutionary Algorithms for Data Mining and Knowledge, *Advances in Evolutionary Computation, 2002* – Citeseer

Goldberg, D.E.(1989). *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley

Mehrotra, K., Mohan, C. K. & Ranka, S. (1997). *Elements of Artificial Neural Networks*, MIT Press, Cambridge, MA.

Siebel, N. T., Krause, J., & Sommer, G. (2007). Efficient Learning of Neural Networks with Evolutionary Algorithms, *Lecture Notes in Computer Science* , Springer

Stepniewski, S.W. & Keane, A. J. (2006). Topology design of feedforward neural networks by genetic algorithms ,*Lecture Notes in Computer Science*, Springer

Wang, C. And Principe, J. C. (1999). Training Neural Networks With Additive Noise in The Desired Signal, *IEEE Transactions on Neural Networks*.
http://www.eia.gov/dnav/pet/hist/LeafHandler.ashx?n=PET&s=RBRTE&f=D
http://www.gold.org/investment/statistics/gold_price_chart