An evolutionary approach to neural network design applied to sunspot prediction

J. Hakkarainen, A. Jumppanen, J. Kyngäs and J. Kyyrö

Department of Computer Science,
University of Joensuu

Abstract - *In this paper we describe a neural network simulator which uses evolutionary optimization for the selection of an optimal architecture and learning parameters for a MLP neural network. We also present some simulation results of sunspot prediction problem.*

Keywords - evolutionary algorithms, neural networks, sunspot prediction

1 INTRODUCTION

Multilayer perceptron (MLP) neural networks have been used for solving many interesting problems like speech generation (Sejnowski 1986), classification of sonar signals (Gorman 1988) and identification of genetic facies (Cardon, Hoogstraten and Davies 1991).

MLP networks can be trained for solving a particular problem by using a simple learning algorithm like *backpropagation* (Rumelhart, Hinton and Williams 1986). The training is an automatic process, but unfortunately the selection of the network structure is not. The input features and the number of hidden nodes must be carefully chosen: if the input feature set is incorrect or if the network has too few hidden nodes the learning process fails. On the other hand, if the network has too many nodes in the hidden layer(s) it can overfit the training data.

Usually the network structure is selected by using trial-and-error method. Recently, however, *evolutionary algorithms* have been considered as a promising method for selecting the optimal network structure.

Evolutionary algorithms consist of optimization and search techniques like *Genetic Algorithms* (Goldberg 1989), *Genetic Programming* (Koza 1992) and *Evolutionary Programming* (Fogel 1966).

In the literature there exists many combinations of evolutionary algorithms and neural networks. Yao (1992) gives an overview of the work done in this area. Evolutionary algorithms have been used at least for (1) weight initialization (Masters 1993), (2) network training (Montana and Davis 1989), (3) network structure and learning parameters selection (Harp and Samad 1991), and (4) learning rule selection (Chalmers 1990). In this paper we describe the *ANNE* (Artificial Neural Network Evolution) simulator, which by using evolutionary techniques can automatically choose the appropriate number of hidden nodes and the proper learning parameters. In addition, the simulator can perform genetic input feature selection proposed by Masters (1993).

This paper is organized as follows: In the next section we describe the algorithms used in ANNE simulator. We have applied the simulator to sunspot prediction. The results are presented in Section three. In Section four we present our conclusions.

2 THE SIMULATOR

In the simulator, feedforward networks with one hidden layer are used. In theory, this kind of networks are capable of approximating any Borel measurable function to any desired degree of accuracy (Hornik, Stinchcombe and White 1989).

Algorithm 1 presents the evolutionary algorithm used in the simulator. The goal of the algorithm is to find a network that minimizes the error in the test data. Each network has a “chromosome” that defines what input features the network uses, the number of hidden nodes, the activation function, the error function minimized in training and the learning algorithm and parameters.

Both logistic function

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

and hyperbolic tangent

$$f(x) = \tanh(1.5x) \tag{2}$$

can be used as an activation function. The possible error functions are the *mean squared error* (MSE)

$$E = \frac{1}{n}\Sigma(a_i - t_i)^2, \tag{3}$$

the *Kalman-Kwasny error* (Kalman & Kwasny 1991, Masters 1994)

$$E = \frac{1}{n}\sum\frac{(a_i - t_i)^2}{1 - a_i^2} \tag{4}$$

and *cross entropy error* (Masters 1994)

$$E = \frac{1}{n}\sum\left(t_i \log\left(\frac{t_i}{a_i}\right) + (1 - t_i)\log\left(\frac{1 - t_i}{1 - a_i}\right)\right). \tag{5}$$

First a population pool of networks with weights selected from a uniform probability distribution between -1 and 1 is created. The networks are trained n epochs using the backpropagation or *scaled conjugate gradient* (Møller 1993) algorithm. The networks that give the worst results for the test data are removed from the population. The best networks are probabilistically chosen for reproduction so that the networks giving better results have a higher probability of being selected. The created new networks inherit parameters randomly from their parents (uniform crossover). A mutation can change the parameters with a small probability p. Finally, the new networks are trained the same amount as their parents. The whole process is repeated until the performance of the best network is sufficient enough.

The maximum number of hidden nodes h_{max}, the mutation probability p, the parent selection criteria, and the range of the learning parameters are user selectable parameters.

The simulator also contains a pre-processor, which can be used for input data scaling, and a post-processor, which converts the results back to the original range.

ALGORITHM 1:

Create m random networks
i:= 0
repeat
    Train each network n epochs with the training data
    Evaluate the performance of each network against the test data
    Remove the weakest networks
    Reproduce children for the best networks
    Apply mutation operator to the children
    i:= $i + n$
    Train the children i epochs
until *end-condition*

3 SUNSPOT PREDICTION

To test our simulator we applied it to the sunspot prediction problem, which is a commonly used benchmark problem both in neural network and statistics literature.

In sunspot prediction we try to predict the yearly numbers of sunspots. The predictions are based on previous years sunspot numbers. Because the sunspot time series has an irregular cycle it is very difficult to predict (Box an Jenkins 1976).

We used the yearly average sunspot numbers from years 1700-1920 as training data and the sunspot numbers from years 1921-1955 as test data. Because the simulator uses the test data to evaluate the performance of the networks, we used an additional data set of sunspot numbers from years 1956-1979 for validation.

In order to compare our algorithm to random search we ran two simulations: one in the normal way described in Algorithm 1 (NN_Norm) and another in which the children were created randomly, so that they did not inherit their parameters from their parents (NN_Random).

The populations used in the simulations consisted of 250 networks. After training each generation, 50 children were reproduced for the 200 best performing networks. The size of the hidden layer was allowed to vary between 2-5 neurons, the learning rate parameter between 0.01-0.50 and the momentum between 0.00-0.95.

After the populations had evolved for 38 generations the training was stopped and the performance of the populations best networks was evaluated. We compared the root mean squared errors, defined as

$$\mathrm{RMSE} = \sqrt{\frac{\sum (a_i - t_i)^2}{n}} \qquad (6)$$

for the training data, test data (1921-1955) and the validation data (1956-1979) (see Table 1).

As can be seen from Table 1, the network created by the evolutionary method outperformed the random search network. Both networks picked all of the inputs (lags 1 through 12) to be significant.

The evolutionary network ended up to with a network with 5 hidden neurons and the hyperbolic tangent as the activation function. In contrast the random search network ended up with a network with 4 hidden units and the logistic function as the activation function.

TABLE 1. RMSEs of the networks.

	NN_Norm	**NN_Random**
Training set	11.384	13.162
Test set	9.784	13.507
Validation set	20.968	27.774

To test how well the best neural network model actually predicts future sunspot numbers, we selected two models from the literature for comparison purposes.

The first model is identified by Ghaddar and Tong (Tong 1995) back in 1980. They actually fitted two models, one for raw data and one for a square-root transformed data. We chose the model built on square-root transformed data, because Ghaddar and Tong favored it and it produces more accurate predictions. As far as we know it is the best statistical model ever built to predict sunspot numbers.

The other model is a neural network model created by Weigend, Huberman and Rumelhart back in 1990 (Weigend & al. 1990). It is a feedforward neural network model with one hidden layer consisting of three sigmoidal neurons and a single linear output neuron. They claim that their network outperforms the non-linear statistical model of Ghaddar and Tong. However they have compared their model only against the raw data model of Ghaddar and Tong.

We compared these three models with two different error measures. The RMSE and the *average relative variance* (ARV) defined by Weigend et al. (1995) as

$$\mathrm{ARV} = \frac{1}{n}\frac{1}{\sigma^2}\sum (a_i - t_i)^2 . \qquad (7)$$

The RMSE tells how much the predictions differ from the observed values on an average. To be more precise it is the standard deviation of the prediction error.

The ARV is an interesting error measure. Because the dynamic range of the data is removed (division by the variance), different prediction sets can be compared to obtain the order of prediction difficulty. However Weigend et al. have computed the ARV values for different prediction sets by substituting the true variance of the prediction set by a constant 1535. The reason for this is not clear; removing the variance term from the formula leaves us with the mean squared error.

We chose the RMSE as an error measure because in our opinion it is easy to understand and it penalizes big absolute prediction errors. The ARV is chosen for two reasons: first, Weigend et al. (1990) had used it, and second, to find out which of the time intervals is hardest to predict in respect to the time interval's variance. We divided the sunspot numbers time series into five parts as follows:

1. years 1700-1920 (training set)
2. years 1921-1955 (test set)
3. years 1956-1979 (validation set 1)
4. years 1980-1994 (validation set 2)
5. years 1956-1994 (validation set 3).

We compared the ability of the models to predict one step ahead and N steps ahead. One step ahead means that the model predicts one value with observed sunspot numbers as inputs. This procedure is iterated as many times as is the length of the period to predict. In the N steps ahead procedure the predictions are based on predictions. So observed sunspot numbers are only presented to the model in the beginning of the predicting period.

Table 2 shows the RMSEs of the model's lag one predictions and Table 3 shows lag N predictions for different data sets (the graphs of the predictions are shown in Appendices I and II). The RMSEs of training set tells how good the model fits the training/identification data. The RMSE:s of test set are not comparable, because we have used the corresponding data in our simulator to monitor the performance of the networks. The RMSEs of validation sets can be directly compared between different models.

In lag one prediction our neural network model and the Tong model seems to be almost equally good. When the validation set 2 is used our neural network model seems to be better but that illusion is cleared out when we look at the results of validation set 3. The Weigend network, however, is worse than our network and the Tong model in every data set.

In lag N prediction every model in every data set seems to have poor predicting abilities. The network trained by Weigend et al. seems to be the best predictor in terms of RMSE.

TABLE 2. RMSEs of lag one predictions.

	MODEL		
Data set	**Evol NN**	**TONG**	**WEIGEND**
training	11.384	12.264	**11.258**
test	**9.784**	10.762	11.584
validation 1	20.968	**19.647**	23.323
validation 2	**16.567**	20.271	22.098
validation 3	**19.394**	19.889	22.860

TABLE 3. RMSEs of the lag N predictions.

	MODEL		
Data set	**Evol NN**	**TONG**	**WEIGEND**
test	26.250	30.010	**24.486**
validation 1	33.290	35.441	**27.576**
validation 2	**22.605**	39.484	31.224
validation 3	39.439	43.905	**34.476**

Tables 4 and 5 present the ARV values of lag one predictions (Table 4) and of lag N predictions (Table 5). The ARV values can be examined to find out which of the data sets is hardest to predict for each model in respect to its variance. The smaller the ARV value the easier the corresponding data set is to predict.

From Table 5 can be observed that the longer the time interval to predict the worse is the result. This is obvious when viewing the ARV value of validation set 3 (years 1956-1994) against other time intervals.

TABLE 4. ARV values for lag one predictions.

	MODEL		
	Evol NN	**TONG**	**WEIGEND**
training	0.110	0.127	**0.107**
test	**0.057**	0.069	0.080
validation 1	0.144	**0.127**	0.179
validation 2	**0.103**	0.154	0.183
validation 3	**0.129**	0.136	0.179

TABLE 5. ARV values of the lag N predictions.

	MODEL		
Data set	**Evol NN**	**TONG**	**WEIGEND**
test	0.412	0.538	**0.358**
validation 1	0.364	0.412	**0.250**
validation 2	**0.192**	0.586	0.366
validation 3	0.534	0.661	**0.408**

TABLE 6. Variances of the data sets.

Data set	Variance
train	1183
test	1674
validation 1	3047
validation 2	2662
validation 3	2914

Table 6 shows the variances of the sunspot numbers in different data sets. The smaller the variance is the easier it is to predict the sunspot numbers from that data set.

We used also our network for predicting sunspot numbers for years 1995-2000. The predictions are shown in Table 7.

TABLE 7. Predicted sunspot numbers for years 1995-2000.

Year	Sunspots
1995	17.9
1996	20.3
1997	35.7
1998	78.7
1999	133.2
2000	136.2

4 CONCLUSION AND DISCUSSION

In this paper we have demonstrated a neural network simulator that applies evolutionary optimization to the selection of neural network architecture and input features. In our simulations the simulator was capable in finding the relevant input features and a suitable network architecture better than a random search. However, the simulation with a single processor computer is computationally very expensive. In order to really take advantage of the evolutionary methods, the simulator should be implemented by using parallel hardware.

Although the results of the sunspot prediction problem seem to point out the superiority of the evolutionary network, this is not necessarily true. Future research is required to find out the effect of the population size and the mutation probability to the results. Some of our simulations indicate that if the search space is "small" or the size of the population is big, there is little or no advantage of using proposed evolutionary methods in the training process. This is though a hypothesis still and will receive our future interest.

We are also unhappy with the fact that usually when different predicting methods are presented the only way they are compared is one or two error measures or maybe some error measure that the author have invented. Sometimes it almost looks like that the error measures are chosen so that they make the presented model to look good. Also, as Tong (1995) points out, the predicting models can be tuned to perform well with the "unseen" data. Tong suggests that only "genuine predictions" should be used, but this leads to a situation where no comparisons between models can be made. We just have to trust on the honesty of the researchers.

REFERENCES

Box G. E. P. and Jenkins G. M., "*Time Series analysis: Forecasting and Control*," Holden-Day Inc., San Francisco, 1976.

Cardon H., Hoogstraten R. and Davies P., "A Neural Network Application in Geology: Identification of Genetic Facies," *Artificial Neural Networks* (Eds. Kohonen T. Mäkisara K., Simula O. and Kangas J.), pp. 809-813. Elsevier, 1991.

Chalmers D. J., "The evolution of learning: an experiment in genetic connectionism," *Proceedings of the 1990 Connectionist Models Summer School*, pp. 81-90. Morgan Kaufmann, 1990.

Fogel L. J., Owens A. J. and Walsh M. J., "*Artificial Intelligence through Simulated Evolution*," John Wiley & Sons, 1966.

Goldberg D. E., "*Genetic Algorithms in Search, Optimization and Machine Learning*," Addison-Wesley, 1989.

Gorman R. P. and Sejnowski T. J., "Analysis of Hidden Units in Layered Network Trained to Classify Sonar Targets," *Neural Networks*, 1, pp. 75-89, 1988.

Harp S. A. and Samad T., "Genetic Synthesis of Neural Network Architecture," *Handbook of Genetic Algorithms* (Ed. Davis L.), pp. 203-221. Van Nostrand Reinhold, 1991.

Hornik K., Stinchcombe M. and White H., "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, 2, pp. 359-366, 1989.

Kalman, B. L. and Kwasny S. C., "Why Tanh? Choosing a Sigmoidal Function," *International Joint Conference on Neural Networks*, Baltimore, 1992.

Koza J. R., "*Genetic Programming: On the Programming of Computers by Means of Natural Selection*," The MIT Press, 1992.

Masters T., "*Practical NN Recipes in C++*," Academic Press, 1993.

Masters T., "*Signal and image processing with neural networks*," John Wiley & Sons, Inc, 1994.

Montana D. and Davis L., "Training feedforward neural networks using genetic algorithms," *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 762-767. Morgan Kaufmann, 1989.

Møller M. F., "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning," *Neural Networks*, 6, pp. 525-533, 1993.

Rumelhart D. E., Hinton G. E. and Williams R. J., "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing* (Eds. Rumelhart D. E., McClelland J., and the PDP Research Group). MIT Press, 1986.

Sejnowski T. J. & Rosenberg C. R., "NETtalk: a parallel network that learns to read aloud," *Technical Report JHU/EECS-86/01*. The Johns Hopkins University Electrical Engineering and Computer Science, 1986.

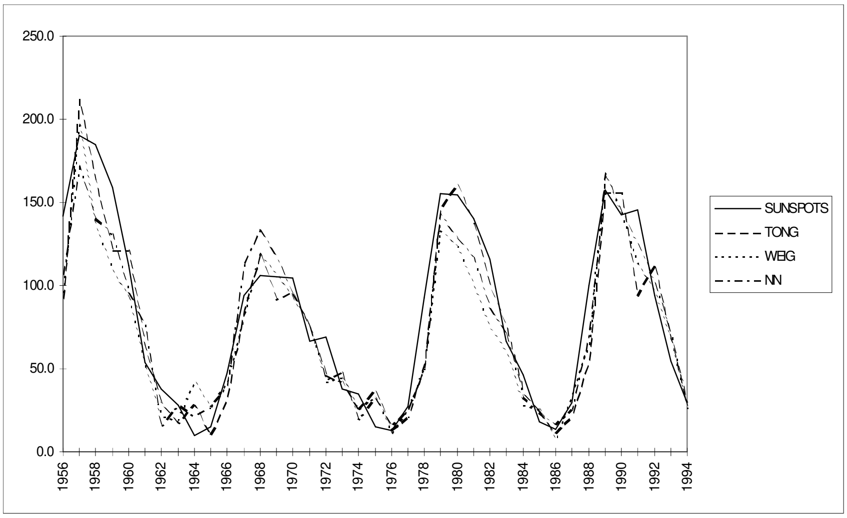
Tong H., "Non-linear Time Series: A Dynamical System Approach," Oxford University Press Inc., New York, 1995.

Tong H., "*Threshold Models in Non-linear Time Series Analysis*," Lecture Notes in Statistics (Eds. D. Brillinger, S. Fienberg, J. Gani, J. Hartigan, and K. Krickeberg), Vol 21. Springer-Verlag NewYork, 1983.

Weigend A. S., Huberman B. A. & Rumelhart D. E. , "Predicting the Future: A Connectionist Approach", International Journal of Neural Systems, Vol 1, 3, pp. 193-209, 1990.

Yao X., "A Review of Evolutionary Neural Networks," *Technical Report*. Commonwealth Scientific and Industrial Research Organization, 1992.

APPENDIX I -The lag one predictions of the different models.



APPENDIX II - The lag *N* predictions of different models.