NSC Project Reports
NSC 87-2415-H-004-011
From 86/08/01 to 87/07/31
Genetic Programming and Evolutionary Artificial Neural Nets in Financial Engineering

Shu-Heng Chen
AI-ECON Research Group
Department of Economics
National Chengchi University
Taipei, Taiwan 11623
TEL: 886-2-2387308
FAX: 886-2-2391344
E-mail: chchen@nccu.edu.tw

Key Words: Artificial Neural Networks, Evolutionary Computation, Genetic Programming, Tick-by-Tick Data, Forecasting.

1 General Background

Since the pioneering work by White (1988), the application of *artificial neural networks* (ANNs) to finance has enjoyed an exponential growth in research and publications. The evidences accumulated over the last decade indicate that the success of the financial application of ANNs crucially depends on the design of the ANN.

Let us first consider the *structure* of the ANN. It is well known that *nonlinear ARMA time series* can be more efficiently approximated by *recurrent neural nets* than by their *layered feedforward* counterparts. (Kamijo and Tanigawa, 1990; Lee and Park, 1992; Kaun and Liu, 1995). Also, with the presence of structural switches in time, it may be more appropriate to use *modular neural networks* (Kimoto, Asakawa, Yoda, and Takeoka, 1990). The many successful applications of *radial basis neural networks* in finance further suggests the relevance of this special class of artificial neural nets to finance (Hutchinson, Lo and Poggio, 1994).

Once the structure has been decided, the next complicated task to deal with in the design of the ANN is the *architecture*. In this area, one of the most frequently discussed issues is the *number of hidden nodes*. It is widely accepted by finance people that the number of hidden nodes are closely related to the issue of *overfitting*, and that a proper choice of the number of hidden nodes (*hidden layer size*) can enhance the generalisation capability of ANNs. Different techniques have been tried by researchers to control hidden layer size, among them, *cross validation* (Moody and Utans, 1995), *early stopping* (Kimoto and Asakawa, 1990; Hoptroff, Bramson and Hall, 1991), *complexity regularization* (Weigend, Huberman and Rumelhart, 1992), *construction-and-destruction algorithms* (Jang and Lai, 1994).

In contrast, the significance of *transfer functions* has received less attention among finance people in their design of ANNs. While the relevance of the transfer function to the design of ANNw has been shown earlier by Mani (1990), Lowell and Tsoi (1992) and, recently, by Sebald and Chellapilla (1998), its role in the financial domain has not been well documented. Even though the radial basis network has been promoted, the choice of *sigmoid, hyper tangent* or *radian basis functions* seems to be based largely on the *hunch* of the researcher rather than the performance of each function. In their option pricing model, Hutchinson, Lo and Poggio (1994) compared the performance of the radial basis function and the sigmoid function and did not find significant difference between them. However, in a similar application, Chen and Lee (1998) provided some evidences of the superiority of the hyper tangent function over the sigmoid function.

Apart from hidden layer size and transfer functions, other potential contributing factors, such as the *learning rate*, *momentum*, and *training methods*, have also attracted the attention of some finance people, though their significance is far less clear.

The brief review above roughly summarizes the list of factors which have been considered relevant to a successful application of ANNs in the financial domain. Needless to say, the ANN space based on such a list is incredibly large. While experts' prior knowledge or domain-specific knowledge may reduce the size of this space, financial theory can help us little in this regard. At its best, financial theroy can only guide us in the selection of *structure* (recurrent or feedforward net) and *inputs*, and it can hardly say anything about the hidden layer size or transfer functions.

In addition to financial theory, statistical theory and information theory have also been employed to solve the design problem. In most cases, they are applied in such a manner that only one factor can be addressed at a time. For example, when the only design issue is the control of hidden layer size, the ANN can be designed in accordance with a statistical or an information-theoretic approach, such as the frequently quoted statistics, such as the *Bayesian information criterion* (BIC) and *Predictive Stochastic Complexity* (PSC). However, if, in the meantime, the choice of transfer functions is also a problem, then this approach may no longer be applicable.

So, when both finance and statistics reach their limits, some search procedures must be taken, and this is where *evolutionary artificial neural nets* come into play. Evolutionary ANNs (**EANNs**) can be considered a combination of ANNs and *evolutionary search procedures*. A prominent feature of EANNs is that they can evolve towards the *fittest* in a task environment *without outside interference*, and thus eliminate the tedious trial-and-error work of *manually* finding an optimal (fittest) ANN for the task about which *little* prior knowlegde is available. Yao (1993) provided an survey of the development in this research area. Yao distinguished among three kinds of evolution in EANNs, i.e., the evolution of *connection weights*, *architectures*, and *learning rules*.

However, like many other "solutions", an EANN is not a panacea, and, what is worse, it may create more problems before it can solve any. One of the major issues in EANNs is the *representation problem*. While encoding *connection weights* is straightforward, encoding the *architecture* and the *learning rule* are daunting tasks. As Yao (1993) stated:

> Trying to develop a universal representation scheme which can specify any kind of dynamic behaviours of an EANN is clearly impractical, let alone the prohibitive long computation time required to seach such a learning rule space.

Therefore, what he further suggested is that it may not be a good strategy to use evolutionary computation at all levels of the evolution of an ANN. So, the question left is: *at what level, can evolutionary computation be helpful for the design of EANNs in finance?*

The last few years have seen a series of financial applications of EANNs. Margarita (1991) applied a genetic search to the weights of a recurrent network for the trading of the FIAT shares in the Milan Stock Exchange. In Dorsey, Johnson and Mayer (1995), the GA was found to perform

well when optimizing NNs. Sexton, Johnson and Dorsey (1995) also found the GA-optimized NN to outperform the BPNN when testing out-of-sample, thereby addressing the problem of overfitting. Harrald and Kamstra (1998) used *evolutionary programming* to replace the more familiar backpropagation method to fine tune the connection weights of feedforward nets for forecasting volatility. White (1998) showed that a Genetic Adaptive Neural Network (GANN) is able to approximate, to a high degree of accuracy, the complex, nonlinear option-pricing function used to produce the simulated option prices.

While these studies clearly evidenced the promising feature of using evolutionary computation in the design of ANNs, in Yao's categorization they are all concerned with the lowest level of evolution, namely, *connection weights*. Other dimensions of the design of an artificial neural net, such as the number of hidden layers, number of hidden nodes, inputs, and transfer functions, have not been tackled. Therefore, the purpose of this research is *to extend the current financial applications of EANNs to a higher level of evolution, and to evaluate its relevance.*

2 Summary: Publications, Softwares and Websites

To do this, this research project develops its own software, called *GPNN*, stands for *Genetic Programming in Neural Networks*. This software is now available from our website:
http://econo.ncnu.edu.tw/ai/staff/sch/Software.htm

We have applied this software to the test of three sets of problems. First, it was applied to forecasting stock returns. The result entitled **"Evolutionary Artificial Neural Networks and Genetic Programming"** (with C.-C. Ni) was published in

- G. D. Smith, N. C. Steele and R. F. Albrecht (eds.) (1998), *Artificial Neural Nets and Genetic Algorithms*,, Springer. pp. 397-400.

Second, a follow-up research is to apply it to option pricing, and is concluded with two papers. The first one entitled **"Option Pricing with Genetic Programming"** (with C.-H. Yeh and W.-C. Lee) was published in

- Koza, et al (eds.) (1998), *Proceedings of the Third Annual Genetic Programming Conference*, Morgan Kaufmann. pp. 32-37.

The second one entitled with **"Hedging Securities with Genetic Programming"** (with W.-C. Lee and C.-H. Yeh) was published in

- Nakhaeizadeh, G. and E. Steurer (eds.) (1998), *ECML'98 Workshop Notes on Application of Machine Learning and Data Mining in Finance*, Technishce Universitat Chemnitz, pp. 140-151.

It has been further submitted to *International Journal of Intelligent Systems in Accounting, Finance and Management*. Thirdly, this technique is applied to *high-frequency financial data*. Lots of nonlinear econometric techniques are required for the purpose of preprocessing data. All relevant econometric programs are also available from our websites:
http://econo.ncnu.edu.tw/ai/staff/csh/course.htm#Financial Economics
The result entitled **"Would Evoultionary Computation Help for Designs of Artificial Neural Nets in Financial Applications"** (with C.-F. Lu) is to be published in

- Greenblatt, S. A. (eds.), *Soft Computing in Financial Markets*, International Computer Science Conentions.

Table 1: Stylised Facts of DM/US Returns: 6/3/98, 3799 Observations.

Procedure	Result	Procedure	Result
Skewness	-0.0196	Kurtosis	4.3912
Jargue-Bera	306.6182*	AR(1)	-0.4202
BDS	23.997	BDS[AR.(1)]	15.871*
BDS[MA(1)]	13.798*	Hurst Exponent	0.38

The BDS tests reported here were conducted by taking $\sigma = 1$ and m(embedding dimension)=5. BDS[AR(1)] refers to the BDS test on the residuals after being prewhiten by an AR(1) filter, and similar for BDS[MA(1)].

The paper listed above will also been put in my website for the interested reader to download. http://econo.nccu.edu.tw/ai/staff/csh/vitachen.htm
Some results which have not been published is briefly summarized as follows.

3 Data Description

Our data set is composed of intradially foreign-exchange-rate returns for the USD/DEM. The main source of this data set is the interbank spot prices published by Dow Jones in a multiple contributors page (the TELERATE page). This covers markets worldwide and 24 hours a day. Those prices are quotations of bid and ask prices and not actual trading prices. Furthermore, they are irregularly sampled and therefore termed as *tick-by-tick prices*. The specific date of the intraday data employed is June 3, 1998, which is free from the *weekend* or the *Monday effect*. There are 3,800 observations in the original series. By taking log and difference, we have 3,799 observations in return series.

We first conducted a series of statistical procedures to examine the existence of stylized facts in this dataset. The procedures and results are exhibited in Table 1. From Table 1, we can see that this intraday data holds many stylised facts, such as being *not IID*, *not normally distributed*, and *first-order negative correlated*. (Zhou, 1996) The data indicates the possible existence of hidden nonlinearity and hence opens a door for the application of ANNs.

4 Experimental Design

In the following, we will discuss whether ANNs can help extract nonlinear structures from the data. And if so, would we further benefit by using evolutionary computation to guide the design of ANNs. This leads us to consider three classes of models. The first one is *the random-walk model*. The second is a class of back-propagation ANNs with prespecified architectures. The third is a class of ANNs whose architectures are determined by evolution. For brevity, the second class of models is entitled BPNNs, and the third EANNs. As mentioned earlier, a higher level of evolution is what we have in mind. Therefore, our EANNs include evolution of the number of inputs, the number of hidden layers, the number of hidden neurons, transfer functions (*sigmoid* and *tanh*), the learning coefficients and momentum. We used the first 2,500 observations as the training set, the next 500 observations as the validation set, and the last 800 observations as the testing set. The software *NueroGenetic Optimizer* (version 2.5) is used to implement the computation.

The relevant parameters for the BPNNs are given in Table 2. Eight settings of BPNNs are considered in this study. Four different input sets are used. There are $\Omega_1 (\equiv \{r_{t-1}\})$, $\Omega_2 (\equiv \{r_{t-1}, \sigma^2_{t-1}\})$, $\Omega_{15} (\equiv \{r_{t-1}, r_{t-2}, ..., r_{t-15}\})$, and and $\Omega_{16} (\equiv \{r_{t-1}, r_{t-2}, ..., r_{t-15}, \sigma^2_{t-1}\})$, where r_{t-i} is the return

Table 2: Designs of BPNNs

	BP1	BP3	BP5	BP7	BP2	BP4	BP6	BP8
# of Inputs	1	15	2	16	1	15	2	16
# of Hidden Layers	1							
# of Hidden Nodes	1-10							
Initial Weights	$\pm$ 0.3							
Transfer Function	tanh for the hidden layer linear for the output layer				sigmoid for the hidden layer linear for the hidden layer			
Learning Rate	0.1 or 0.8 for the hidden layer 0.1 or 0.4 for the output layer							
Momentum	0.1 or 0.6 for the hidden layer 0.1 or 0.2 for the output layer							
Fitness Function	Mean Square Error (MSE)							

Due to Lapedes and Farber (1987, 1988), the transfer functions for the output layers are all linear. Initial weights are set ranging from -0.3 to 0.3. The learning rate is set at either 0.1 or 0.8 for the hidden layer, and 0.1 or 0.4 for the output layer. Momentum is set at 0.6 and 0.1 for the hidden layer, and 0.2 or 0.1 for the output layer. The energy function is defined by mean square error.

for the $t-$ ith tick, and σ^2_{t-1} refers to the volatility of the last tick. σ^2_{t-1} is dervied by the simple moving-average formula. More precisely,

$$\sigma^2_{t-1} = \begin{cases} \frac{\sum_{i=1}^{5}(r_{t-i}-\bar{r}_5)^2}{4}, & \text{if } \Omega_2 \text{ is used,} \\ \frac{\sum_{i=1}^{15}(r_{t-i}-\bar{r}_{15})^2}{14}, & \text{if } \Omega_{16} \text{ is used.} \end{cases} \tag{1}$$

where

$$\bar{r}_5 = \frac{\sum_{i=1}^{5} r_{t-i}}{5} \tag{2}$$

$$\bar{r}_{15} = \frac{\sum_{i=1}^{15} r_{t-i}}{15} \tag{3}$$

The number of hidden nodes is set to be from 1 to 10. For each number of hidden node and a set of the learning rate and momentum, an ANN is randomly generated, and is trained by using *backpropagation method* and *the early stopping rule*. We then use the *validation set* to test the inferred networks, and the best among these 10 ANNs is kept for the post-sample forecasting competition. This procedure is applied to each setting, and in the end, eight BPNNs were selected out of 80 BPNNs. The similar procedure was taken to generate 8 EANNs, each distinguished by its design of evolution. Table 3 summarizes their differences.

Notice that the "Max # of Inputs" assigned to each setting refers to the number of inputs potentially being available. The actual number of inputs used is finallized by evolution. The input sets considered here are the same as BPNNs. The only difference is that now we are letting genetic serach to decide which inputs should be included. Similarly, for Max # of Hidden Layers", in the case where there are two hidden layers, the number of hidden layers actually used is determined by evolution. Given the parameters chosen in Table 3, both the weights, the architectures and learning coefficients will be determined by the genetic search driven by the *NeuroGenetic Optimizer*, Version 2.5.

Table 3: Designs of EANNs

Setting	GA1	GA2	GA3	GA4	GA5	GA6	GA7	GA8
Max # of Inputs	1	15	2	16	1	15	2	16
Max # of Hidden Layers	1				2			
Transfer Functions	tanh, sigmoid, linear							
Learning Rate	[0.1, 0.8] for the hidden layer [0.1, 0.4] for the output layer							
Momentum	[0.1, 0.8] for the hidden layer [0.1, 0.4] for the output layer							
Population Size	50							
Generation	50							
Crossover Rate	0.6							
Mutation Rate	0.001							
Fitness Function	Mean Square Error (MSE)							
Selectin Mode	Roulette Wheel Selection							

The range of the learning rate is set at *[0.1, 0.8] for the hidden layer, and [0.1, 0.4] for the output layer.* **The range of momentum is set at** *[0.1, 0.8] for the hidden layer, and [0.1, 0.2] for the output layer.*

5 Experimental Results

Based on the testing procedures described above, we ran a forecasting competition among 16 ANNs (8 BPNNs and 8 EANNs) and the random-walk (RW) model. The performance criteria are mean absolute error, mean squared error, Theil's U, hit ratio and Sharpe ratio. Together, these five criteria enable us to see differences, if any, among these models, not only from a statistical standpoint, but from an economic perspective. Furthermore, several statistical tests were run to see whether the differences are significant. The tests employed are the Granger-Newbold test, the sign test, the Pesaran-Timmermann test, and the Diebold Mariano test.

Due to the space limit, we only summarise a few major experimental results here. *First, can ANNs beat the random-walk model in terms of intraday data?* The answer is *positive*. All our 16 ANNs beat the RW model in all the five criteria above. This result is interesting in its own right. It indicates that, to beat the RW, what we need is a reasonable, rather than an optimal, design of the ANN. *Second, are the differences found significant?* The answer is "*not always*". In terms of the Granger-Newbold test and the sign test, the result is clear. All 16 ANNs beat the RW model at the 1% significance level. However, in terms of the Pesaran-Timmermann test, only two out of these sixteen (BPNN4 and EANN7) outperformed the RW model significantly. As to the Diebold-Mariano test, none of these ANNs beat the RW significantly. Thus, the superiority of nonlinear models, such as ANNs, over the RW in forecasting, is not statistically evident as far as high-frequency data are concerned.

Now comes the core issue: *can evolution help the design of ANNs?* The results are *mixed*. While in the criteria MAE, MSE, Theil's U, EANN8 took the lead, it is the BPNN4 that was the top performer in the hit ratio and the Sharpe ratio. If we look further down the ranking, we will see that more BPNNs ranked higher than the EANNs. In fact, in all five criteria, the worst performers belonged to the EANN family. Therefore, at the current stage, what we have found can be concluded as follows.

- ANNs in general can be anticipated to outperform, at least *numerically*, the RW in the

intraday data. To get this superior performance, what we need is only a reasonable design of the ANN rather than an optimal one.

- While the design can be crucial for the financial application of ANNs, evolutionary computation may not help that much. So far, we have seen that evolutionary computation can be helpful only up to the level of *connection weights*. Beyond that, there is no evidence for any significant improvement.

References

[1] Chen, S.-H. and W.-C. Lee (1998), "Pricing Taiwan Call Warrants with Artificial Neural Netwroks," *paper submitted to 1999 International Joint Conference on Neural Networks* (**IJCNN'99**).

[2] Dorsey, R. E., J. D. Johnson and W .J. Mayer (1994), "A Genetic Algorithm for the Training of Feedforward Neural Networks," in J. D. Johnson and A. B. Whinston (eds.), *Advances in Artificial Intelligence in Economics, Finance, and Management*, Vol. I, JAI Press. pp. 93-111.

[3] Harrald, P. G. and M. Kamstra (1997), "Evolving Artificial Neural Networks to Combine Financial Forecasts," *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 40-52.

[4] Hoptroff, R. C., J. Bramson, and T. J. Hall (1991), "Forecasting Economic Turning Points with Neural Nets," *Proceding of International Joint Conference on Neuarl Nets*, Vol. 1, pp. 347-352.

[5] Hutchinson, J. M., A. W. Lo, and T. Poggio (1994), "A Nonparametric Approach to Pricing and Hedging Derivative Securities Via Learning Networks," *Journal of Finance*, Vol. XLIX, No. 3, pp. 851-889.

[6] Jang, G.-S. and F. Lai (1994), "Intelligent Trading of an Emerging Market," in Deboeck, G. J. (ed.), *Trading on the Edge: Neural, Genetic, and Fuzzy Systems for Chaotic Financial Markets*, Wiley.

[7] Kamijo, K.-I., and T. Tanigawa (1990), "Stock Price Pattern Recognition: A Recurrent Neural Network Approach," *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 1215-1221.

[8] Kimoto, T., K. Asakawa, M. Yoda, and M. Takeoka (1990), "Stock Market Prediction System with Modular Neural Networks," *Proceding of International Joint Conference on Neural Networks*, Vol. 1, pp.1-6.

[9] Kuan, C.-M. and T. Liu (1995), "Forecasting Exchange Rates Using Feedforward and Recurrent Neural Networks," *Journal of Applied Econometrics*, Vol. 10, pp 347-364.

[10] Lee, C. H. and K. C. Park (1992), "Prediction of Monthly Transition of the Composition of Stock Price Index Using Recurrent Back-Propagation," *Artificial Neural Network 2*, pp. 1629-1632.

[11] Lovell, D. R. and A. C. Tsoi (1992), "The Performance of the Neocognitron with Various S-cell and C-cell Transfer Functions," Intelligent Machines Laboratory, Department of Electronic Engineering, Univ. of Queensland.