Advanced Neural Network Training Methods for Low False Alarm Stock Trend Prediction

Emad W. Saad, Danil V. Prokhorov, and Donald C. Wunsch II
Applied Computational Intelligence Laboratory
Department of Electrical Engineering, Texas Tech University
Lubbock, TX 79409-3102
Tel.: (806) 742-1266, Fax.: (806) 742-1245
E-mail: SaadE@ttu.edu

ABSTRACT

Two possible neural network architectures for stock market forecasting are the time-delay neural network and the recurrent neural network. In this paper we explore two effective techniques for the training of the above networks, i.e. conjugate gradient algorithm and multi-stream extended Kalman filter. We are particularly interested in limiting false alarms, which correspond to actual investment losses. Encouraging results have been obtained when using the above techniques.

1. Introduction

Stock forecasting involves complex interactions between market-influencing factors and unknown random processes. Neural networks (NN) are used in this application for their ability to model complex nonlinear systems based on sample data. We are interested in predicting profit opportunities of a given stock. Here we define a profit opportunity as an increase of 2% or more in the stock price within one month (22 working days). Earlier we successfully demonstrated that such short-term predictions are achievable with an acceptable false alarm rate using probabilistic and time-delay neural networks [1]. The probabilistic network was trained to make direct trend predictions whereas the time-delay network was taught to do iterative price forecasts from which the trend prediction could be calculated. A false alarm is a predicted increase followed by no actual increase. False alarms are the most important factor to eliminate, because if the system is used for investment decisions, false alarms correspond to losses.

Training obviously requires some form of memory being present in the network architecture. We use Time Delay Neural Networks (TDNNs) and Recurrent Neural Networks (RNNs) trained by conjugate gradient method and multi-stream Extended Kalman Filter, respectively.

2. Time-Delay and Recurrent Neural Networks

Data of daily stock price can be treated as a time series. If we have the daily prices of a certain period, they can be fed to a neural network which predicts the future price of the first day following the end of the known period. We repeat this process iteratively to predict the price 22 working days ahead. TDNN is known to be an appropriate architecture for time series prediction, due to its internal memory [2, 3]. For instance, the simplest TDNN is the multi-layer perceptron which uses the input tapped-delay line. Each predicted price is then compared with the closing price at the beginning of the period and if the price on any of the 22 days increased 2% or more, then we say that there is a profit opportunity for the current day. Thus we are making a 22 day sliding window forecast. In a similar application of a neural network on multi-step ahead prediction of sunspot series, Weigend et al. has found that it outperformed a Threshold Autoregressive Model [10].

RNNs used in this study are multi-layer perceptrons with fully feedback-connected hidden layers. As in the case of the probabilistic network [1], we have trained RNNs to make direct predictions of the stock market trends.

3. Conjugate Gradient Training

Conventional training of TDNN consists of minimizing a cost function, which is usually the mean square error of the network. The backpropagation algorithm computes derivatives of the cost function with respect to the network weights. Weights are then updated using different learning rules, such as the steepest descent. Unfortunately, using the antigradient as the learning direction is not the best choice because minimization along one direction may be spoiled by a subsequent minimization along another direction, thus slowing down the training process [4, 5].

The conjugate gradient learning method has been demonstrated as a powerful industrial tool [11]. It consists of choosing "conjugate directions" of minimization that don't interfere with each other, and performing a line minimization of the cost function along that direction. The algorithm can be summarized as follows:

1. Calculate the direction of descent using:

$$d_{k+1} = -g_{k+1} + \beta_k d_k \ , \qquad (1)$$

where d is the direction of descent, and g is the gradient vector calculated using backpropagation. There are several variations of how to compute the scalar β. The simplest is the Fletcher-Reeves version

$$\beta_k = g^T{}_{k+1}\, g_{k+1} \,/\, g^T{}_k g_k \qquad (2)$$

We used the following update formula:

$$d_{k+1} = -g_{k+1} + ((1+y^T{}_k y_k / p^T{}_k y_k)\, p^T{}_k g_{k+1} / p^T{}_k y_k - y^T{}_k g_{k+1} / p^T{}_k y_k)\, p_k + (p^T{}_k g_{k+1} / p^T{}_k y_k)\, y_k \qquad (3)$$

suggested by Shanno [6], where $y_k = g_{k+1} - g_k$, $p_k = w_{k+1} - w_k$ and w is the weight vector. This allows the use of an approximate line minimization, thus reducing the number of forward propagations required every pass, and accelerating the training process.

2. Minimize the error function along the direction d.
3. Go to step 1 and repeat until a convergence criterion is satisfied.

4. Multi-Stream Extended Kalman Filter-Based Training of RNNs

Among many methods of training RNNs, the Extended Kalman Filter-based (EKF-based) training has been shown to be an effective and powerful technique [7]. It belongs to the class of second-order training methods, and it adapts weights of the network in a pattern-by-pattern fashion accumulating important training information in approximate error covariance matrices and providing individually adjusted updates for each of the network's weights.

When there is a large data base including a variety of operational conditions (e.g., different predominant trends of the market), a batch update of weights is advantageous over a pattern-by-pattern update since the network learns to simultaneously minimize error on a whole batch of patterns taken from different regions of the data base. To combine efficiency of the EKF-based training with a batch-like update, without violating consistency between the weights and the approximate error covariance matrices P_j, the multi-stream training approach was first proposed in [8].

Algorithmically, the major difference between the ordinary EKF-based training and its multi-stream version lies in the way of updating weights of the network. Instead of updating weights immediately after proper derivatives were obtained and stored in a set of matrices H_j, as in the original method, the single weights' update is delayed until all N streams are processed, and each matrix H_j becomes N times larger. Thus, the global scaling matrix A also grows in size N times for each of its dimensions, and it is computed by the following formula

$$A = [\eta I + \sum H_j^T \; P_j \; H_j]^{-1} \tag{4}$$

where η is a learning rate, and the summation runs through the whole set of matrices H_j. Despite the increased computational burden associated with a necessity of inverting the larger size matrix A, one gains a significantly faster training and better generalization.

We have used the multi-stream EKF-based training of RNNs for predicting profit opportunities of several stocks. To be able to affect a degree of conservatism of the network's predictions, it is desirable to incorporate penalizing factors in the error measure. In training the RNN, we have used a penalty for false alarms stronger than that for missed opportunities.

5. Results

Some of our results are given in the Table 1. For instance, for IBM stock TDNN correctly predicted 37 profit opportunities out of 117 possible with 8.1 % of false alarms vs. 78 predictions of RNN with 1.3 % of false alarms. Figure 1 shows best results for Apple, using TDNN. Figure 2 shows best results for IBM, using RNN. In each case, a reasonable number of profit opportunities have been identified, with a very low false alarm rate. Some stocks gave better results compared to others due to their higher predictability [9].

	# of profit opportunities, total	# of predictions TDNN/ RNN	% of false alarm TDNN/RNN
Apple Computer	94	30/ 36	0/ 2.8
IBM	117	37/ 78	8.1/ 1.3
Motorola	108	29/ 39	13.8/ 5.1

Table 1: Results for AAPL, IBM & MOT stocks

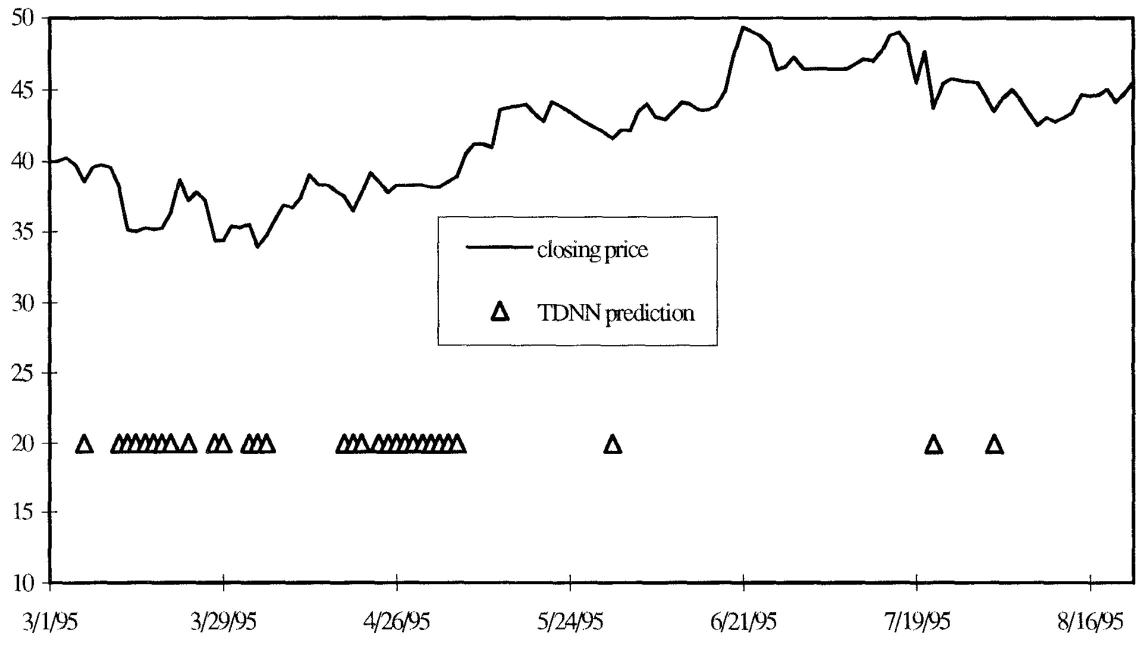

Figure 1. TDNN predictions for Apple Computer.

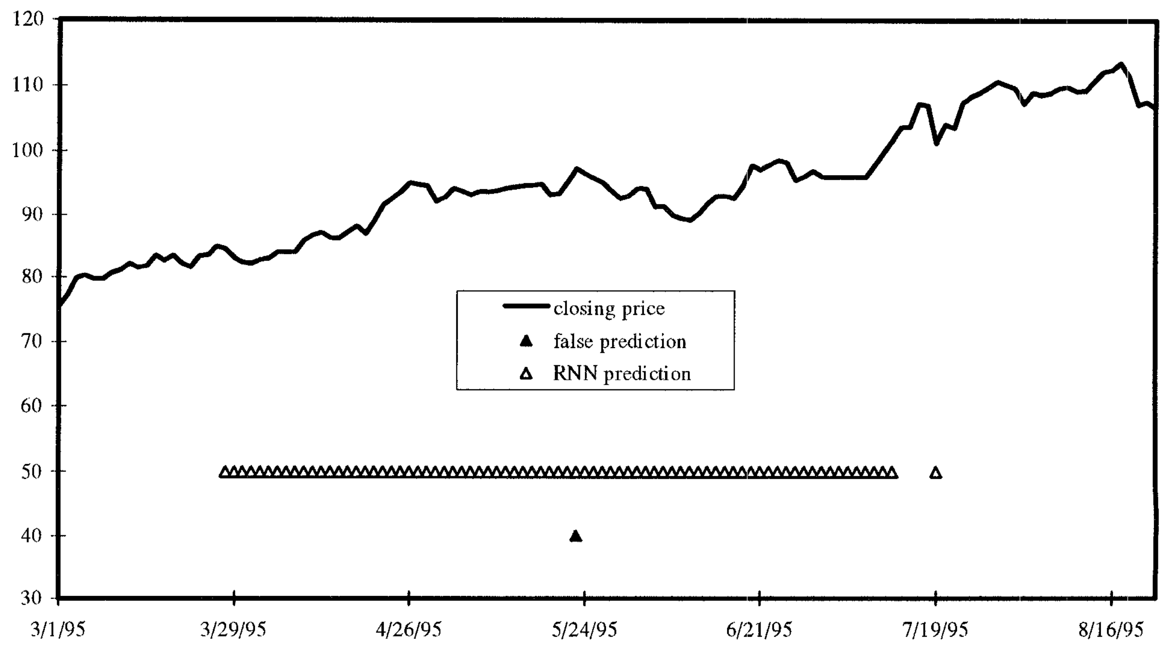

Figure 2. RNN predictions for IBM.

6. Conclusion

Two different neural network architectures have been tested on predicting stock trend. A very low false alarm rate has been obtained, which is the most important factor in minimizing losses. The use of two advanced training techniques, conjugate gradient and extended Kalman filter has greatly accelerated training and allowed to obtain a better performance. Efficient training of TDNN and RNN with the above methods increases the potential of these neural networks in financial forecasting.

References

[1] H. Tan, D. Prokhorov, and D. Wunsch, "Probabilistic and Time-Delay Neural Network Techniques for Conservative Short-Term Stock Trend Prediction," *Proc. of the World Congress on Neural Networks*, Washington, DC, July 1995.

[2] W. Kreesuradej, D. Wunsch, and M. Lane, "Time-Delay Neural Network for Small Time Series Data Sets," *Proc. of the World Congress on Neural Networks*, San Diego, CA, June 1994.

[3] E. Wan, "Time Series Prediction by Using a Connectionist Network with Internal Delay Lines," *Time Series Prediction: Forecasting the Future and Understanding the Past* (A.S. Weigend and N.A. Gershenfeld, eds.), pp. 195-217. Reading, MA: Addison-Wesley.

[4] S. Haykin, "Neural Networks A Comprehensive Foundation," McMillan, 1994.

[5] W. Press et al., "Numerical Recipes in C, The Art of Scientific Computing," Second Edition, Cambridge University Press, 1994.

[6] D. Shanno, "Conjugate Gradient Methods With Inexact Searches," *Mathematics of Operations Research*, vol. 3, No. 3., 1978.

[7] G. Puskorius, and L. Feldkamp, "Neurocontrol of Nonlinear Dynamical Systems with Kalman Filter-Trained Recurrent Networks," *IEEE Trans. Neural Networks*, March 1994.

[8] L. Feldkamp, and G. Puskorius, "Training Controllers for Robustness: Multi-stream DEKF," *Proc. of the World Congress on Computational Intelligence*, Orlando, FL, June-July 1994, pp. 2377-2382.

[9] E. Saad, D. Prokhorov. and D. Wunsch, "Comparative Approach to Financial Forcasting Using PNN, TDNN and RNN," to be submitted to *IEEE Trans. Neural Networks.*

[10] A. Weigend, B. Huberman and D. Rumelhart, "Predicting The Future: A connectionist Approach," *International Journal of Neural Systems*, vol. 1, pp. 193-209, 1990.

[11] J. Hornell, "Industrial Experiences with Neural Networks," *Proc. of the 6th Oklahoma Symposium on Artificial Intelligence,* Tulsa, OK, Nov. 11-12, 1992, pp. 127-133.