Constrained Formulations and Algorithms for Stock-Price Predictions Using Recurrent FIR Neural Networks

Benjamin W. Wah and ***Minglun Qian***
Department of Electrical and Computer Engineering
and the Coordinated Science Laboratory
University of Illinois, Urbana-Champaign
1308 West Main Street, Urbana, IL 61801, USA
E-mail: {wah, m-qian}@manip.crhc.uiuc.edu

Abstract

In this paper, we develop new constrained artificial-neural-network (ANN) formulations and learning algorithms to predict future stock prices, a difficult time-series prediction problem. Specifically, we characterize stock prices as a non-stationary noisy time series, identify its predictable low-frequency components, develop strategies to predict missing low-frequency information in the lag period of a filtered time series, model the prediction problem by a recurrent FIR ANN, formulate the training problem of the ANN as a constrained optimization problem, develop new constraints to incorporate the objectives in cross validation, solve the learning problem using algorithms based on the theory of Lagrange multipliers for nonlinear discrete constrained optimization, and illustrate our prediction results on three stock time series. There are two main contributions of this paper. First, we present a new approach to predict missing low-pass data in the lag period when low-pass filtering is applied on a time series. Such predictions allow learning to be carried out more accurately. Second, we propose new constraints on cross validation that can improve significantly the accuracy of learning in a constrained formulation. Our experimental results demonstrate good prediction accuracy in a 10-day horizon.

I. Introduction

In this paper we study the following stock-price prediction problem. Given a sequence of daily closing prices $R(t)$ of a stock and the corresponding low-pass version $S(t)$ of $R(t)$, predict $S(t_0 + h)$ at *prediction horizon* h from the current time t_0 using only the history of closing prices $R(t_0), R(t_0-1), R(t_0-2), \ldots$ For simplicity, we only consider univariate time-series predictions in this paper.

The prediction problem as defined is difficult for the following reasons. First, the time series is noisy and non-stationary, making it difficult to use past information to predict future trends. We show in Section II that noise does not contribute to prediction accuracy and needs to be removed. Hence, the problem involves the prediction of a smoothed time series that may lag behind actual price changes. Second, many factors leading to price fluctuations cannot be captured precisely or may be too numerous and too difficult to be modeled. As a result, the prices of a single stock, represented by a univariate time series, may not be enough to accurately predict its future prices.

We measure prediction quality by two metrics. The first metric is the widely used *normalized mean square error* ($nMSE$) defined as follows:

$$nMSE = \frac{1}{\sigma^2 N} \sum_{t=t_0}^{t_1} (o(t) - d(t))^2, \qquad (1)$$

where σ^2 is the variance of the true time series in period $[t_0, t_1]$, N is the number of patterns tested, and $o(t)$ and $d(t)$ are, respectively, the network and desired outputs at time t.

The second metric is the *hit rate* defined as follows. Let $D(t+h) = \text{sign}(S(t+h) - S(t))$ be the actual direction of change for $S(t)$, and $\hat{D}(t+h) = \text{sign}(\hat{S}(t+h) - \hat{S}(t))$ be the predicted direction change. We call a prediction for horizon h a *hit* iff $\hat{D}(t+h) \times D(t+h) > 0$. The hit rate $H(h)$ is defined to be

$$H(h) = \frac{\left|\left\{t | D(t+h)\hat{D}(t+h) > 0, t = 1, \cdots, n\right\}\right|}{\left|\left\{t | D(t+h)\hat{D}(t+h) \neq 0, t = 1, \cdots, n\right\}\right|} \qquad (2)$$

where $|E|$ represents the number of elements in set E. Figure 1 illustrates the definition. Assume that $S(t)$ lags behind and is only available up to $t = t_0 - m$. $\hat{S}(t)$, the predicted price in $t \in [t_2, t_3]$, is lower than $\hat{S}(t_0)$, the predicted price at t_0. This prediction is a hit because the actual price at t is also lower than $S(t_0)$. In contrast, $\hat{S}(t')$, the predicted price in $t' \in (t_1, t_2)$ is not a hit because it is lower than $\hat{S}(t_0)$, whereas $S(t')$ is higher than $S(t_0)$.

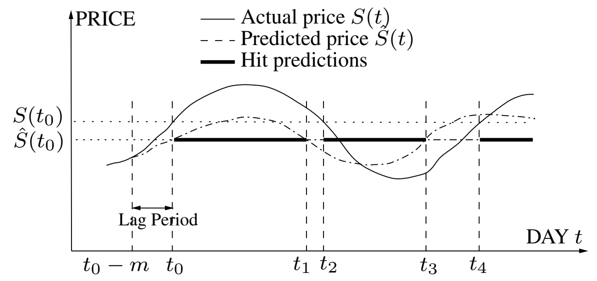

Figure 1: An illustration of hits in predictions.

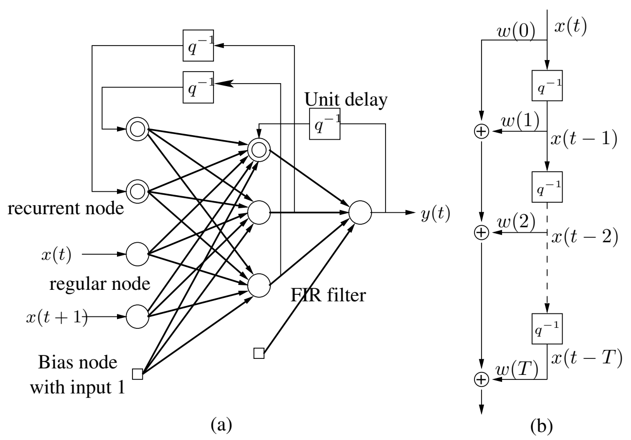

Figure 2: Structure of a three-layer RFIR: a) Recurrent FIR neural network; b) FIR filter. In a), double concentric circles indicate recurrent nodes, other circles are non-recurrent nodes, and small boxes are bias nodes with constant input 1. q^{-1} represents one unit time delay.

The hit rate is very useful when a trading decision is based solely on whether the predicted future price goes up or down as compared to the current price (Saad *et al.* 1998). Other metrics may be used, depending on the trading strategy. Without loss of generality, new metrics used can be added as constraints in cross validations in our formulation.

Existing models for time-series analysis can be classified into *linear* and *nonlinear models* (Chatfield 2001). Linear models such as Box-Jenkins' *ARIMA* and its variations (Box and Jenkins 1976) and *Exponential smoothing* (Brown 1963) work well for linear time series but fail to model complicated nonlinearity and trends in financial time series. *State-space models* (Aoki 1987) are another class of linear models that represent inputs as a linear combination of a set of state vectors that evolve over time according to some linear equations. In practice, state vectors and their dimensionality are hard to choose (Chatfield 2001). Nonlinear models, such as *time-varying parameter models* (Nicholls and Pagan 1985) and *threshold auto-regressive models* (Tong 1990), generally pre-specify a special nonlinear function to be used. These models are not effective for modeling financial time series because the nonlinear functions are hard to choose. Another class of nonlinear models are *artificial neural-network models* (ANN) that can model processes with unknown dynamics (Haykin 1999). They have been proved to be universal function approximators and do not require inside knowledge on the process under investigation. In this paper we use a recurrent FIR ANN (RFIR) proposed in our early work (Wah and Qian 2001b). As shown in Figure 2, an RFIR ANN is similar to a recurrent ANN except that each connection is modeled by an FIR filter instead of a single synaptic weight. As a result, RFIR combines a recurrent structure in recurrent ANNs (Elman 1990) (RNN) and an FIR structure in FIR ANNs (Wan 1993) (FIR-NN), and can store more historical information than either alone.

This paper is organized as follows. Section II presents the analysis of time series of stock prices and shows that low-pass filtering is needed to remove unpredictable high-frequency components. It further proposes schemes to overcome edge effects in low-pass filtering. Section III describes a constrained ANN formulation. By including new constraints to compensate for edge effects and on errors and hit rates in cross-validation, we apply violation-guided back-propagation to find suitable weights. Finally, Section IV compares the performance of different prediction methods.

II. Preprocessing of Time Series

Time series of most financial applications has been found to be non-stationary, noisy (Zheng *et al.* 1999) and behave like random walks (Hellstrom and Holmstrom 1997). To improve their predictability, they are preprocessed in order to remove unpredictable noise and enhance their stationarity.

A time series is said to be *stationary* if it has (near) constant mean and variance, where stationarity is often tested by computing its autocorrelations (Masters 1995). The autocorrelation of a stationary time series drops rapidly from a significant non-zero value to zero, whereas that of a non-stationary time series stretches out indefinitely. Two widely used methods to transform a non-stationary series to a stationary one are linear de-trending and differencing (Masters 1995), although such methods have not been found to work well on raw time series of stock prices. In particular, differencing may lead to accumulated errors when stock prices are reconstructed from the differenced time series.

On the other hand, unwanted noise can be removed by low-pass filtering or wavelet de-noising. There is a lot of recent interest in using wavelet transforms to decompose financial time series into bands in such a way that each band contains a set of near stationary signals and that the low-frequency band is smooth enough to be converted to stationary series. Unfortunately, de-noising introduces lags in the smoothed data because it uses future data in its computations. Such edge effects are not desirable because they reduce the number of most recent patterns that are critical for predictions, thereby introducing uncertainties in both $\hat{S}(t_0+h)$ and $\hat{S}(t_0)$ in computing the hit rate.

Recently, a redundant Á *Trous* wavelet transform was proposed for de-noising instead of traditional decimated wavelet transforms (Zheng *et al.* 1999). It was argued that such transforms provide *shift invariance* and that the information at time t in each resolution reconstructs the original signal directly at time t with lags. Hence, the decomposition by wavelet transforms helps identify the properties of the signals at each resolution level. As an illustration, Figure 3 shows the autocorrelations (AC) of the closing prices of IBM, as well as the high-frequency channels w_1, W_2, W_3, W_4 and low frequency channel c_4 resulted from the Á *Trous* transform. The curves show that $R(t)$ and c_4 are non-stationary, whereas w_1 and w_2 are stationary.

Besides stationarity, we also need to consider the lag of a time series. A decomposed stationary time series that lags behind the original series is not useful for prediction unless future data beyond the lag period is correlated to the filtered series. Consider Table 1 that shows the lag and the number of autocorrelation coefficients that are larger than 0.5 for each resolution level of the decomposed IBM and Microsoft

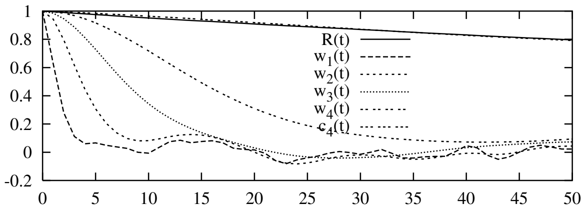

Figure 3: Autocorrelations (AC) of IBM's closing prices and their five decomposed channels using the Á *Trous* wavelet transform

Table 1: Low correlations for most channels beyond the lag period, based on the wavelet transforms in Figure 3.

Decomposed Signal	Days with $ACF > 0.5$ IBM	Days with $ACF > 0.5$ MSFT	Lag
w_1	1	1	0
w_2	3	2	2
w_3	7	6	6
w_4	14	$\boxed{15}$	14
c_4	$\boxed{50+}$	$\boxed{50}$	30

(MSFT) closing prices. The results show that, except for c_4, signals beyond the lag period are weakly correlated to signals available in the decomposed time series, making it difficult to predict beyond the lag period.

Due to a lack of correlation between signals in the high-frequency components and those beyond the lag period and the fact that high-frequency components generally have small magnitudes, we only consider in this paper the prediction of the low-frequency components of daily closing stock prices. Specifically, we use a 20-tap low-pass filter designed by Matlab function $firls(20, [0\ 0.1\ 0.2\ 1], [1\ 1\ 0\ 0])$. (Other low-pass filters will give similar results.) The low-pass filter will incur a 10-day lag in the smoothed time series. Further, for the same reason as in (Zheng *et al.* 1999), we do not use differencing to improve the stationarity of the low-frequency component because we do not want to introduce cumulative errors when stock prices are reconstructed from the differenced time series.

As mentioned before, another critical issue to be addressed in a low-pass time series is the edge effects incurred. Existing schemes handle these effects by extending the raw data $R(t)$ into the future in order to obtain low-pass data up to current time t_0. Some of these common techniques include *wrap-around*, *mirror extension*, *flat extension*, and *zero-padding* (Masters 1995) and are illustrated in Figure 4.

For example, suppose we use a 20-tap filter, and the current time is at t_0. Low-pass data is only available up to time $t_0 - 10$ because a 20-tap low-pass filter needs raw data $R(t_0 - 20)$ to $R(t_0)$ in order to generate $S(t_0 - 10)$. To obtain the missing low-pass data $S(t_0 - 9)$ to $S(t_0)$, flat extension assumes that future raw data $R(t_0 + h) = R(t_0)$ for all $h = 1, 2, \cdots, 10$ before the 20-tap low-pass filter is applied to obtain $S(t_0 - 9)$ to $S(t_0)$. Figure 4 clearly shows that, when a trend is present in the raw data, wrap-around and zero-padding do not work well because they may cause abrupt jumps at the transition point at t_0. In contrast, mirror and flat extensions work well in part of the lag period.

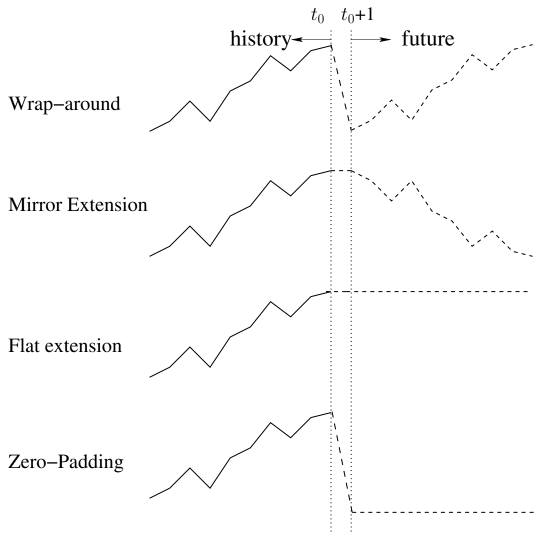

Figure 4: Four approaches for handling edge effect by extending raw data beyond t_0 in order to feed them to the low-pass filtering process. Solid lines represent raw data up to t_0, and dashed lines illustrate extensions to raw data made from $t_0 + 1$ and beyond.

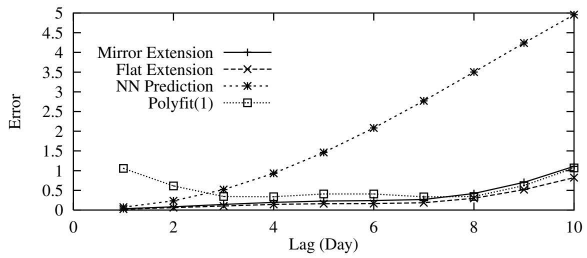

Figure 5: Average errors of IBM's closing prices between April 1, 1997, and March 31, 2002, on four approaches for handling edge effects ($m = 10, q = 7$).

Figure 5 shows that both methods have small average errors for the first seven estimated low-pass points in the lag period with respect to true low-pass data, and flat extensions perform slightly better. Other results (not shown) have similar behavior. This is a surprising result, as mirror extensions have traditionally been viewed as better.

We have also studied the approximation of raw data points in the lag period by a low-order polynomial curve using $polyfit$, a polynomial fitting procedure in Matlab. The objective is to find the coefficients of a polynomial function that minimizes the mean squared errors over the lag period between the raw and fitted smoothed data. Figure 5 shows that the average errors achieved by polynomial fits of degree one are generally larger (fits of higher degrees give worse errors). Note that the errors are particularly large at the beginning of the lag period because the fitted curve is not constrained to have small errors at the transition point.

Yet another method studied is to train an ANN using low-pass data available in order to predict the low-pass data in the lag period. Assuming a lag period of size m, we used a constrained ANN formulation but without the constraints on cross-validation proposed in Section III, trained the ANN to perform one-step predictions using patterns up to $t_0 - m$, and applied the ANN to perform iterative predictions on data in the lag period. The results in Figure 5 show clearly that such an approach performs poorly and justify the need for developing more powerful formulations.

Among all the methods tested, flat extensions achieve the smallest average errors for the first seven days of the lag period, but have considerably larger errors in the last three (Figure 5). Therefore, in our experiments, we use flat extensions to generate seven new training patterns (low-pass data) for the first seven days of the lag period, but include an additional constraint on the raw data in the last three (described in the next section).

III. ANNs for Predicting Filtered Stock Prices

The prediction problem studied in this paper is complex due to the non-stationarity of the time series, multiple objective measures that may not be in closed form, and missing training patterns in the lag period. As a result, it cannot be solved by conventional ANN training methods that perform local searches of a single unconstrained closed-form objective function. In this paper, we adopt a constrained formulation and training strategy we have developed recently (Wah and Qian 2001a), and propose new constraints on raw data in the lag period and on cross validations. By providing new information in constraints that leads a trajectory to reduced constraint violation, a search can overcome the lack of guidance in traditional unconstrained formulations when a trajectory is stuck in a local minimum of its weight space.

As proposed in (Wah and Qian 2001a), we introduce a constraint on the output error of each training pattern:

$$p_i^p(w) = (o(i) - d(i))^2 \le \tau_i^p, \quad 1 \le i \le W, \qquad (3)$$

where $o(i)$ and $d(i)$ are, respectively, the i^{th} actual and desired (target) outputs, τ_i^p is the error tolerance, and W is the window size of the time series used in learning.

We formulate the training objective to be the minimization of the sum of squared one-step prediction errors over a window of training patterns in a recurrent FIR ANN (RFIR) shown in Figure 2. We use the one-step error instead of the more complex iterative prediction error over a horizon because it allows gradients to be computed easily by back-propagation. The complex iterative prediction errors are formulated as constraints in cross validations.

Next, we introduce new constraints to limit errors from multiple validation sets in cross validations. In traditional approaches, cross validations involve the computation of a performance measure on a set of patterns available in training. Since the measure used in cross validations is the same as that in training, patterns used in cross validations and training must be different. The approach of formulating errors in cross validations as constraints not only allows measures that are different from those in learning to be used, but allows patterns in learning and cross validation to be shared. In our work, we avoid overfitting by using small networks along with constraints on iterative validation errors in the validation sets.

Our cross validation simulates as closely as possible the testing process after learning is completed and compares the results of iterative predictions by a trained ANN against

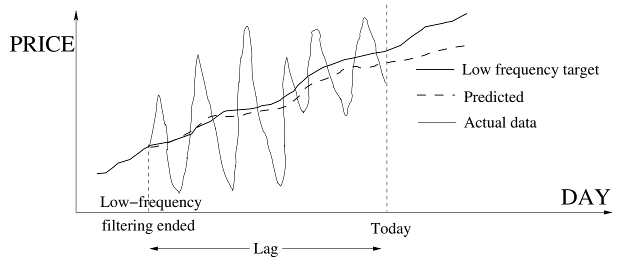

Figure 6: Illustration of constraints on lag period. Outputs of the network in the lag period is constrained to be centered by raw data.

known training patterns. Here, we define a *validation set* to be a collection of $L = m + h$ training patterns, where m is the size of the lag period and h is the horizon to be predicted. Starting from the first pattern in the validation set, we perform a flat extension of the patterns in the first q of the m patterns (described in the last section), perform q single-step predictions using the q extended patterns as inputs, and then perform iterative predictions on the next $h+m-q$ patterns. As training patterns have to be estimated in the lag period and substantial errors are incurred on patterns beyond the lag period, it is more difficult for cross-validation errors to converge. At this point, instead of summing the squares of all the h errors into a single error, as done in (Wah and Qian 2001a), we keep them separately, average the absolute error (MAE) at each horizon over multiple validation sets, and constrain each against a prescribed threshold. Based on the MAE and hit rate metrics defined in Section I, there are $2h$ additional constraints:

$$\begin{array}{rcl} p_e^v(w) & \le & \tau_e^v \\ p_e^r(w) & \le & \tau_e^r \end{array} \qquad 1 \le e \le h, \qquad (4)$$

where $p_e^v(w)$ (resp. $p_e^r(w)$) is the average validation error (resp. residual hit rate $1 - H(e)$) at position e, and τ_e^v (resp. τ_e^r) is the corresponding tolerance. As we expect both validation errors and residual hit rates to increase with an increased horizon, we set τ_e^v and τ_e^r to be monotonically increasing functions with respect to e.

The last constraint to be added involves the $m - q$ patterns in the lag period that are not predicted accurately by flat extensions. As illustrated in Figure 6, the idea is to constrain the predictions of the trained ANN in such a way that its outputs (smoothed data) are centered around the raw data in the entire lag period.

$$p^s = \sum_{t=t_0-m+1}^{t_0-m+q} o(t) - R(t) + \sum_{t=t_0-m+q+1}^{t_0} \hat{S}(t) - R(t) \le \tau^s, \quad (5)$$

where τ_e^s is the tolerance for this error.

Putting all the constraints together, we have:

$$\min_w \quad \sum_{i=1}^{W} \max\left\{(o(i) - d(i))^2 - \tau, 0\right\} \qquad (6)$$

$$\text{s.t.} \quad \begin{array}{l} p_i^p(w) \le \tau_i^p, \quad 1 \le i \le W, \\ p_e^v(w) \le \tau_e^v, \quad 1 \le e \le h, \\ p_e^r(w) \le \tau_e^r \\ p^s = \sum(o(t) - R(t)) + \sum(\hat{S}(t) - R(t)) \le \tau^s. \end{array}$$

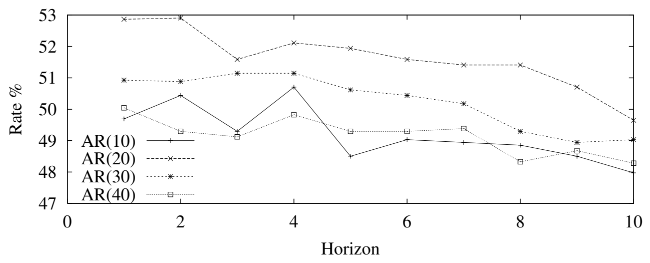

Figure 7: Hit rates of AR on 1,100 consecutive predictions between April 1, 1997, and March 31, 2002, using low-pass filtered data of Citigroup.

Since (6) is a constrained nonlinear programming problem (NLP) with *non-differentiable functions*, it cannot be solved by the traditional back-propagation algorithm or Lagrangian methods that require the differentiability of functions. To address this issue, we apply a new violation-guided back-propagation algorithm (VGBP) we have developed (Wah and Qian 2001a; 2001b) to solve this constrained problem. VGBP works on the Lagrangian function transformed from (6) and searches for discrete-space saddle points based on the *theory of Lagrange multipliers for nonlinear discrete constrained optimization* (Wah and Wu 1999). It does this by gradient descents in the original weight space and ascends in the Lagrange-multiplier space, using an approximate gradient of the objective function found by back-propagation and according to the violation of each constraint. Interested reader can refer to (Wah and Qian 2001b) for details about the VGBP algorithm.

The tolerances τ^p, τ^v, τ^r and τ^s are set by the *relax-and tighten* strategy in VGBP. This strategy is based on the observation that looser constraints are easier to satisfy, while achieving larger violations at convergence, and that tighter constraints are slower to satisfy, while achieving smaller violations at convergence. By using loose constraints in the beginning and by tightening the constraints gradually, learning converges faster with tighter tolerances.

IV. Experimental Results

In this section we compare the performance of our proposed constrained ANN with edge effects handled by flat extension (FE-NN) against four benchmark methods: simple Carbon Copy (CC), auto-regression (AR), ANN similar to FE-NN but without handling edge effects and no cross-validation (NN), and predictions using ideal data in the lag period (IP). CC is easy to implement because it always predicts future data to be same as the most recent available true data. AR was tested using the *TISEAN* implementation (Hegger and Schreiber 2002). We also constructed an ideal predictor (IP) in order to establish an upper bound on prediction accuracy. Based on true low-pass data in the first seven data points in the lag period (instead of predicting them based on flat extensions as in FE-NN), an ANN trained by VGBP is applied to predict the last three missing low-pass data in the lag period and future low-pass data. IP will give an approximate upper-bound accuracy that can be achieved, as it uses seven

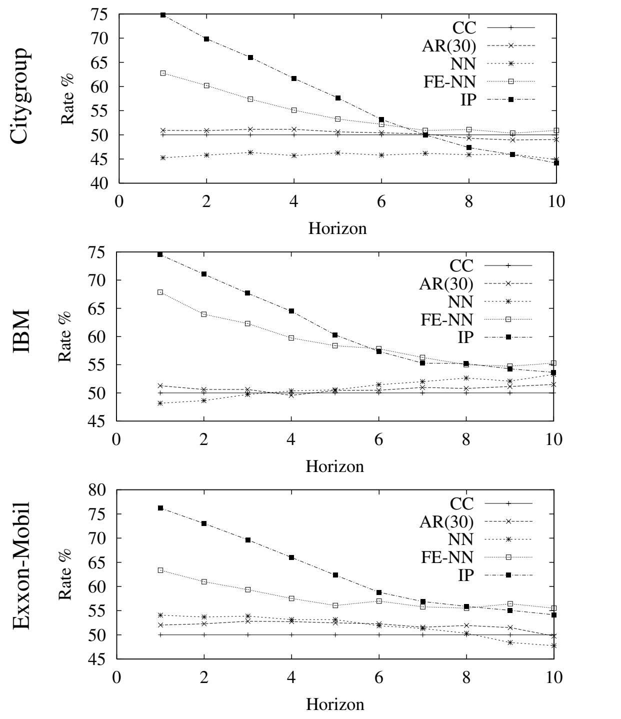

Figure 8: Hit rates on 1,100 consecutive predictions between April 1, 1997, and March 31, 2002, using predictors of Carbon copy (CC), auto-regression with order of 30 (AR(30)), traditional neural network (NN), neural network using constrained formulation coupled with flat extension technique (FE-NN), and ideal predictor (IP).

error-free low-pass data in the lag period that are unavailable otherwise. We tested the aforementioned predictors using three times series based on the closing prices of IBM (symbol **IBM**), Citigroup (symbol **C**) and Exxon-Mobil (symbol **XOM**) from April 1, 1997, to March 31, 2002.

Figure 7 shows the hit rates over different horizons by AR with a variety of orders over 1,100 predictions using low-pass filtered data of Citigroup. As there is a ten-day lag in the low-pass data, AR will need to predict data for ten days in the lag period before predicting into the future. Consequently, its first true prediction starts at the 11^{th} day, and the prediction at horizon h is its $(h + 10)^{th}$ prediction. The results show that pure AR does not provide any useful prediction because its hit rate is always around 50%. Moreover, increasing its order does not improve the accuracy. Similar results have also been observed for the IBM and Exxon-Mobil stock prices. This poor performance may be attributed to the non-stationarity of the time series and the need to predict in the lag period. As AR(30) gives relatively better results, we use it in the remaining experiments on AR.

Figure 8 plots the hit rates for the five predictors. It shows that CC, AR(30) and NN behave like random walks over the ten-day horizon, as they always have around 50% chance to give the correct direction of price changes. On the other hand, our proposed FE-NN can achieve hit rates significantly higher than 50% over small horizons (one to five

Table 2: Normalized mean square errors ($nMSE$s) over 1,100 predictions made between April 1997 and March 2002 using five predictors: carbon copy (CC), autoregression with order 30 (AR(30)), traditional neural network (NN), neural network with constrained formulation and flat extension technique (FE-NN), and an ideal predictor (IP) which uses 7 true low-pass data in the lag period.

Stock	Citigroup			IBM			Exxon-Mobil		
Horizon	1	5	10	1	5	10	1	5	10
C.C	0.061	0.087	0.121	0.085	0.130	0.193	0.122	0.165	0.220
AR(30)	0.055	0.106	0.168	0.068	0.133	0.208	0.136	0.224	0.339
NN	0.086	0.173	0.253	0.095	0.310	0.541	0.183	0.362	0.657
FE-NN	0.010	0.052	0.128	0.013	0.063	0.153	0.019	0.094	0.210
IP	0.002	0.033	0.131	0.003	0.022	0.182	0.005	0.067	0.266

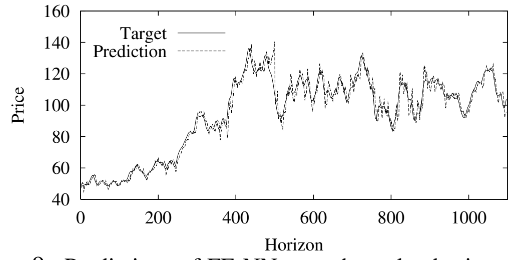

Figure 9: Predictions of FE-NN on a three-day horizon as compared to the actual low-pass data on the 1,100-day closing prices of IBM between April 1, 1997, and March 31, 2002.

days). For these horizons, IP performs better than FE-NN. However, at large horizons (six days and beyond), IP performs statistically the same as FE-NN.

Figure 9 plots the predictions of FE-NN for a three-day horizon, as compared to the actual low-pass data, on the 1,100-day closing prices of IBM between April 1, 1997, and March 31, 2002. The results show that the predictions track well with the actual low-pass data.

Table 2 shows the $nMSE$s of the five predictors over the 1,100 predictions. AR(30) and NN do not perform well because they need to predict iteratively in the ten-day lag period before predicting into the future. FE-NN improves significantly over CC and AR(30), especially for small horizons, and out-performs traditional NN over all horizons. Again, the table shows that FE-NN has errors closest to those of IP over small prediction horizons, and achieves slightly better $nMSE$ at longer horizons. (Note that IP only gives an *approximate* upper bound on prediction accuracy).

References

M. Aoki. *State Space Modeling of Time Series*. Springer-Verlag, Nerlin, 1987.

G. E. P. Box and G. M. Jenkins. *Time Series Analysis: Forecasting and Control, 2nd ed.* Holden-Day, San Francisco, 1976.

R. G. Brown. *Smoothing, Forecasting and Prediction*. Prentice Hall, Englewood Cliffs, NJ, 1963.

Chris Chatfield. *Time-series forecasting*. Chapman & Hall/CRC, Boca Raton, Florida, 2001.

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

S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, NJ, 2 edition, 1999.

R. Hegger and T. Schreiber. The TISEAN software package. *http://www.mpipks-dresden.mpg.de/ tisean*, 2002.

T. Hellstrom and K. Holmstrom. *Predicting the Stock Market*. Technical Report Series IMa-TOM-1997-07, Malardalen University, Vasteras, Sweden, 1997.

T. Masters. *Neural, Novel and Hybrid Algorithms for Time Series Prediction*. John Wiley & Sons, Inc., NY, 1995.

D.F. Nicholls and A.R. Pagan. Varying coefficient regression. In E. J. Hannan, P. R. Krishnaiah, and M. M. Rao, editors, *Handbook of Statistics*, pages 413–449. North-Holland, Amsterdam, 1985.

E.W. Saad, D.V. Prokhorov, and D.C. Wunsch, II. Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. *IEEE Trans. on Neural Networks*, 9:1456–1470, 11 1998.

H. Tong. *Nonlinear Time Series: A Dynamical System Approach*. Oxford University Press, Oxford, 1990.

B. W. Wah and M.-L. Qian. Violation-guided learning for constrained formulations in neural network time series prediction. In *Proc. Int'l Joint Conference on Artificial Intelligence*. IJCAI, 771-776 Aug. 2001.

B. W. Wah and M.-L. Qian. Violation guided neural-network learning fo constrained formulations in time-series predictions. *Int'l Journal on Computational Intelligence and Applications*, 1(4):383–398, December 2001.

B. W. Wah and Z. Wu. The theory of discrete Lagrange multipliers for nonlinear discrete optimization. *Principles and Practice of Constraint Programming*, pages 28–42, October 1999.

E. A. Wan. *Finite Impulse Response Neural Networks with Applications in Time Series Prediction.* Ph.D. Thesis, Standford University, 1993.

G. Zheng, J.L. Starck, J.G. Campbell, and F. Murtagh. Multiscale transforms for filtering financial data streams. *J. of Computational Intelligence in Finance*, 7:18–35, 1999.