International Journal of Information Technology & Decision Making
Vol. 5, No. 4 (2006) 639–658
© World Scientific Publishing Company

World Scientific
www.worldscientific.com

CONSTRAINED FORMULATIONS AND ALGORITHMS FOR PREDICTING STOCK PRICES BY RECURRENT FIR NEURAL NETWORKS

BENJAMIN W. WAH

Department of Electrical and Computer Engineering
and the Coordinated Science Laboratory
University of Illinois, Urbana-Champaign
1308 West Main Street, Urbana, IL 61801, USA
wah@uiuc.edu

MING-LUN QIAN*

Department of Computer Science
University of Illinois, Urbana-Champaign
201 North Goodwin Avenue, Urbana, IL 61801, USA

In this paper, we develop a new constrained artificial-neural-network (ANN) formulation and the associated learning algorithm for predicting stock prices, a difficult time-series prediction problem. We characterize daily stock prices as a noisy non-stationary time series and identify its predictable low-frequency components. Using a recurrent finite-impulse-response ANN, we formulate the learning problem as a constrained optimization problem, develop constraints for incorporating cross validations, and solve the learning problem using algorithms based on the theory of extended saddle points for nonlinear constrained optimization. Finally, we illustrate our prediction results on ten stock-price time series. Our main contributions in this paper are the channel-specific low-pass filtering of noisy time series obtained by wavelet decomposition, the transformation of the low-pass signals to improve their stationarity, and the incorporation of constraints on cross validation that can improve the accuracy of predictions. Our experimental results demonstrate good prediction accuracy and annual returns.

Keywords: Channel-specific low-pass filtering; edge effects; nonlinear constrained optimization; non-stationarity; recurrent FIR neural networks; stock prices; time-series predictions; wavelet decomposition.

1. Introduction

A time series is an ordered sequence of observations in time. It occurs everywhere in our daily life, including financial markets, physical science, and social science. Figure 1 illustrates a time series that depicts the daily closing stock prices of IBM.

*2308 Riva Ridge Road, Montgomery, IL 60538, USA.

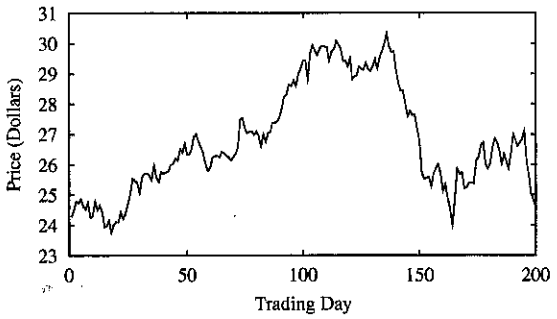


Fig. 1. Daily closing prices for 200 trading days of IBM from January 2, 1990 to October 15, 1990.

The time-series prediction problem entails the development of techniques for predicting future data, based on some temporally and/or spatially distributed historical data. The payoff of time-series predictions can be high, even when a solution of limited accuracy can be found. For example, if an investment firm can predict a certain financial time series slightly better than random guesses, then there are good opportunities to profit. Likewise, if a company can predict its monthly sales more accurately, then it will be in a better position to arrange its business plan.

For time-series predictions to be effective, the historical data in the problem must invariably have some degrees of auto-correlations in time or in space, even though the data may be noisy, uncertain, and incomplete. In general, the underlying dynamics of a time series is either unknown or difficult to represent analytically. For example, there is an obvious up trend between Days 20 and 110 in the time series in Fig. 1 and an obvious down trend between Days 140 and 165. The prediction of these trends, however, may be difficult because the time series is noisy.

A general time series may exhibit nonlinearity, seasonality, non-stationarity, and piecewise chaos. It is nonlinear if the future data to be predicted is a nonlinear function of the past historical data. A time series with dominant periodic components exhibits regular periodic variations. Such behavior can often be found in applications that fluctuate seasonally, such as the annual electricity consumption and merchandise sales. A time series is *piecewise chaotic* if it consists of several regimes in which each corresponds to a chaotic process. It is *stationary* if it has constant mean, finite variance, and its auto-covariance only depends on the distance from the current time. Finally, noise can be present in the entire or some parts of its frequency spectrum. Figure 2 illustrates a noisy non-stationary periodic time series.

In this paper we study the following stock-price prediction problem. Given a sequence of daily average 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.

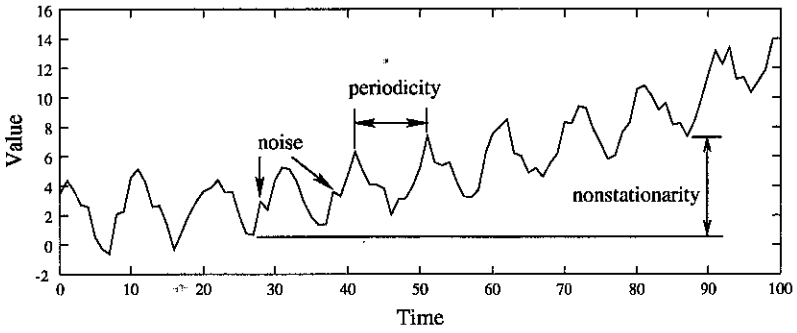


Fig. 2. A time series with a long-term growing trend (non-stationarity), regular alternation of peaks and valleys (periodicity), and small local fluctuations (noise).

The prediction problem as defined is difficult for the following reasons. Firstly, a time series representing the daily prices of a stock is noisy and non-stationary, making it difficult to use past information to predict future trends. We show in Sec. 2 that noise does not contribute to prediction accuracy and needs to be removed. After denoising, the problem entails the prediction of a smoothed time series that may be delayed from the actual price changes. Secondly, many factors leading to price fluctuations cannot be precisely captured or may be too numerous and too difficult to be modeled. As a result, the prices of a single stock in the form of a univariate time series may not be enough to accurately predict its future prices.

The prediction problem studied in this paper involves the following steps. Firstly, we collect and prepare the historical data of the time series to be predicted. This step must be carefully done in order to prevent errors from propagating across samples. One may need to properly select a window of past data for fitting a prediction model. Section 2 presents several pre-processing techniques for removing unpredictable high-frequency noise, improving the stationarity of the low-frequency component, and overcoming edge effects in low-pass filtering.

Next, we select an appropriate model for the time series. Application-specific domain knowledge and proper conditioning of the time series may greatly help select the right model. Existing models for time-series analysis can be classified into linear and nonlinear ones.[1]

Well-known linear models are the Box-Jenkins ARIMA model and its variations, such as AR, MA and seasonal ARIMA.[2] These work well for linear time series but fail to model nonlinear financial time series. Exponential smoothing is a variation of MA that assigns geometrically decreasing weights to older samples in a time series. A smoothed value $S(t)$ can be computed effectively from $R(t)$ by $S(t) = \alpha R(t) + (1 - \alpha)S(t-1)$, where $0 < \alpha \leq 1$. Although the original exponential smoothing cannot handle linear and nonlinear trends, there are extensions for handling linear trends.[1] State-space models[3] are another class of linear models that represent their inputs as a linear combination of a zero-mean noise plus a set of state variables

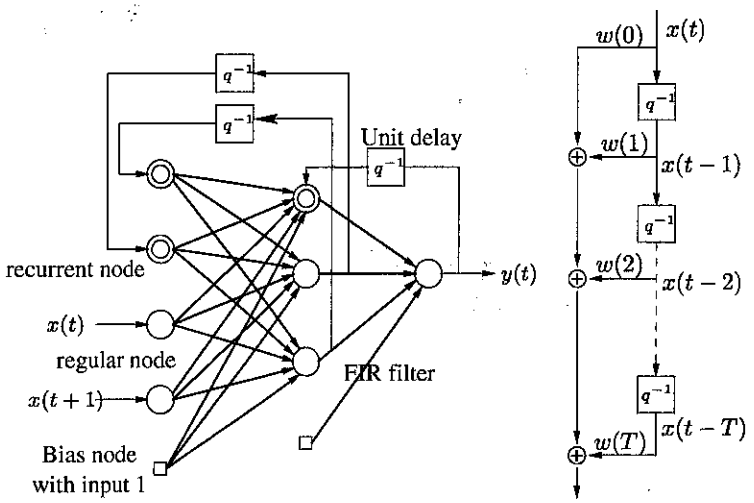
(called state vector) that evolve over time according to some linear equations. In practice, state vectors and their dimensionality are hard to choose, nonunique, and unsuitable for modeling long time series that need a large number of states.[1]

Nonlinear models include nonlinear extensions to ARMA, such as time-varying parameter models[4] and threshold auto-regressive models.[5] These models generally pre-specify a special nonlinear function to be used in regressions and/or moving averages. They are not effective for modeling financial time series whose underlying nonlinear dynamics is ill defined.

Another class of nonlinear models are the artificial neural-network (ANN) models that can model processes with unknown dynamics.[6] An ANN is a nonlinear system for modeling a continuous nonlinear function, without imposing any stochastic or deterministic assumptions on the underlying process that generates the time series. It detects the underlying function by "learning" to predict from historical data.

In this paper, we use a recurrent finite-impulse-response (FIR) ANN (RFIR) developed in our previous work[7] for predicting chaotic time series. As is shown in Fig. 3, 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. It has more powerful modeling ability and can store more historical information because it combines a recurrent structure in recurrent ANNs[8] and an FIR structure in FIR ANNs.[9]

The learning problem is normally solved as an optimization problem that finds a set of optimal or near optimal weights for the model. To avoid getting stuck in suboptimal local minima that do not generalize well in their predictions, one



(a) Recurrent FIR neutral network. (b) FIR filter.

Fig. 3. Structure of a three-layer RFIR. In (a), double concentric circles indicate recurrent nodes, other circles are nonrecurrent nodes, small boxes are bias nodes, and q^{-1} is a unit delay.

can incorporate past training patterns as constraints and identify those patterns that are important for learning and for generalization. Those constraint violations during learning provide valuable guidance when the search gets stuck in a poor local minimum. Section 3 describes a constrained ANN formulation and a violation-guided back-propagation learning algorithm.

Finally, we cross-validate our model on historical data, before applying it in actual predictions. Post-processing of the predictions may be applied to further enhance the prediction accuracy. We evaluate the accuracy of our predictions using the following metrics.

One of the most popular metrics is the widely used normalized mean squared errors ($nMSE$) defined as follows:

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

where σ^2 is the variance of the true time series in $[t_0, t_1]$, N is the number of patterns tested, and $o(t)$ and $d(t)$ are, respectively, the predicted and the desired outputs at t. The measure is not useful for predicting financial time series, because it does not reflect errors in trend predictions, which are important for any investment strategy that tries to maximize a portfolio.

The primary measure we use in this paper is the annual (resp., monthly) percentage return that measures how fast a portfolio is increasing under a certain trading strategy over a year (resp., month). Given $F(t)$ to be the total asset in the portfolio at t, the percentage return for trading strategy $\mathcal{S}$ over $(t_0, t_1]$ is:

$$\mathbb{M}(t_0, t_1) = \left. \frac{F(t_1) - F(t_0)}{F(t_0)} \right|_{\mathcal{S}} \times 100\%. \tag{2}$$

A secondary measure we use in this paper is specific to daily financial data that fluctuates within a range $[R_\ell(t), R_h(t)]$ in Day t. We measure the quality of the predicted time series by the fraction that it is within the daily low/high range. Let $s(t)$ be the predicted time series for $t \in [t_0, t_1]$. Define $\mathcal{A}_0$ to be all those days when the prediction is within the daily low/high range of that day, namely, $\mathcal{A}_0 = \{t \,|\, R_\ell(t) \le s(t) \le R_h(t)\}$. Similarly, define $\mathcal{A}_+$ (resp., $\mathcal{A}_-$) to be all those days when the prediction is higher (resp., lower) than the daily high (resp., low) price of that day, namely, $\mathcal{A}_+ = \{t \mid s(t) > R_h(t)\}$ (resp., $\mathcal{A}_- = \{t \mid s(t) < R_\ell(t)\}$).

To measure how good the predictions track the raw data, we define the *hit ratio* α_0 (resp., *over-hit ratio* α_+ and *under-hit ratio* α_-) to be the ratio of the predictions that fall inside the daily price range (resp., higher than the daily high price, and lower than the daily low price). That is

$$\alpha_0 = \frac{\|\mathcal{A}_0\|}{t_1 - t_0 + 1}, \quad \alpha_+ = \frac{\|\mathcal{A}_+\|}{t_1 - t_0 + 1}, \quad \alpha_- = \frac{\|\mathcal{A}_-\|}{t_1 - t_0 + 1}. \tag{3}$$

Section 4 illustrates the hit ratios and the annual returns of our predictors on a number of stock-price time series.

2. Preprocessing of the Time Series

Financial time series has been found to be non-stationary, noisy,[10] and behave like random walks.[11] To improve its predictability, we preprocess the data in order to remove unpredictable noise and enhance its stationarity.

2.1. *Denoising using wavelet transforms*

Unwanted noise in signals can be removed by filtering.[12] This argument was supported by the Dow Jones theory for stock-price movements[13] that emphasize on primary and secondary trends and that ignore minor trends. Here, *primary trends* are changes that are larger than 20%; *secondary trends* show $\frac{1}{3}$ to $\frac{2}{3}$ relative changes over primary trends; and *minor trends* correspond to noise. However, de-noising by causal filters leads to a smoothed sequence that is a delayed version of the original data. Such *edge effects* are undesirable because the predictor must first correctly predict in the delay period before predicting into the future.

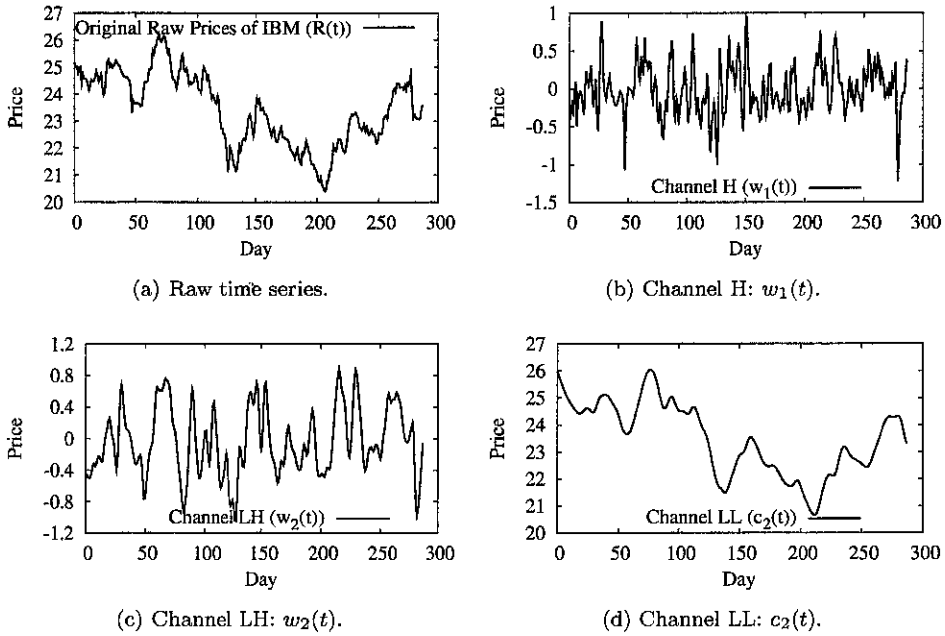
There is growing interest in using wavelet transforms for decomposing financial time series into multiple frequency bands in such a way that each can be handled by some special techniques. Recently, a redundant Á *Trous* wavelet transform was proposed for de-noising instead of the traditional decimated wavelet transforms.[10,14] Without decimation and by using a causal mother filter, the transform provides shift invariance and allows the decomposed information at time t to directly reconstruct the original signal at time t.

The Á *Trous* wavelet transform can be described in the time domain. Suppose a low-pass mother filter with coefficients $f(\ell)$ is used to iteratively decompose $c_0(t) = R(t)$ into M channels: $w_1(t), \ldots, w_M(t)$ and $c_M(t)$, where $w_j(t)$ is the detailed information at resolution level j and $c_j(t)$ is the residue at level j:

$$\begin{aligned} c_j(t) &= \sum_{\ell} f(\ell) c_{j-1}(t - 2^{j-1}\ell) \\ w_j(t) &= c_{j-1}(t) - c_j(t), \quad j = 1, \ldots, M. \end{aligned} \tag{4}$$

A close examination of Eqs. (4) reveals that $c_j(t)$ is a moving average of $c_0(t), c_0(t-1), \ldots, c_0(2^j - 1)$, which means that $c_j(t)$ is a smoothed version of $c_0(t)$. Hence, as with all causal filters, the filtered sequence is a delayed version of the original data.

Figure 4 demonstrates the resulting signals in an Á *Trous* wavelet decomposition using a causal B_3-spline mother filter (with coefficients $f(0) = f(4) = 0.0625$, $f(1) = f(3) = 0.25$, and $f(2) = 0.375$) and $M = 2$. It is clear that the low-pass channel (LL) is very smooth, that the high-frequency channel (H) is very noisy but with much lower magnitude, and that LH is in between in terms of smoothness as well as magnitude. Moreover, LL is delayed from the original sequence by six days.



(a) Raw time series. (b) Channel H: $w_1(t)$.

(c) Channel LH: $w_2(t)$. (d) Channel LL: $c_2(t)$.

Fig. 4. Redundant Á *Trous* wavelet transform of the average daily low/high prices of IBM by a causal B_3-spline mother filter. The coefficients of the filter for each of the three channels are shown in the left panels in Fig. 7. As LL is generated from a causal filter of 13 taps and linear phase, it is delayed from $R(t)$ by six days.

2.2. *Transforming the low-pass signals to improve stationarity*

The decomposition by wavelet transforms helps identify the properties of the signals at each resolution level. The resulting signals, however, differ in terms of their stationarity. In general, the stationarity of a time series can be tested by computing its autocorrelations,[15] where the autocorrelation of a stationary time series rapidly drops from a significant nonzero value to zero, and that of a nonstationary time series stretches out over a large horizon. Figure 5 shows the autocorrelations of

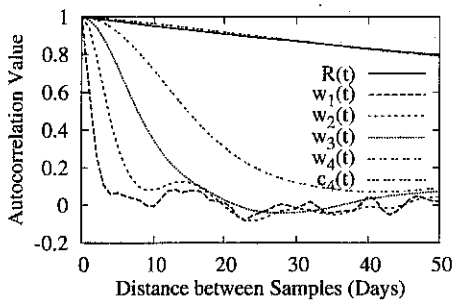


Fig. 5. Autocorrelations of IBM's closing prices and their five decomposed channels using a causal B_3-spline mother filter. All the decomposed channels except $c_4(t)$ are stationary.

the closing prices of IBM, as well as those of the five channels decomposed using a causal B_3-spline mother filter. It shows that $R(t)$ and $c_4(t)$ are non-stationary, and that $w_1(t)$ through $w_4(t)$ are stationary.

Because a nonstationary time series is hard to predict,[7] we need to first transform it into a stationary time series. Two widely used methods are linear de-trending and differencing,[15] although they have not been found to work well on raw stock prices. In particular, differencing may lead to accumulated errors when values are reconstructed from the differenced time series.

To transform a nonstationary time series into near-stationary chaotic time series, we perform the following pattern-wise transformation. Assume the input sequence is $(x(t-W+1),\ldots,x(t))$ and the desired output is $x(t+1)$. We define:

$$\begin{aligned} \mu &= \frac{1}{W}\sum_{i=W-1}^{0} x(t-i), \quad a = \min_{0\le i\le W-1}\{x(t-i)\}, \\ b &= \max_{0\le i\le W-1}\{x(t-i)\}, \quad c = \max\{\mu-a, b-\mu\}. \end{aligned} \tag{5}$$

Then the transformed signals are given by

$$y(t-i) = \frac{x(t-i)-\mu}{c}, \quad \text{for all } i = W-1, W-2, \ldots, 0, -1. \tag{6}$$

The transformation confines the transformed patterns in a window of size W to $[-1,1]$ with a mean of 0. Note that the desired output after transformation may be outside the range $[-1,1]$.

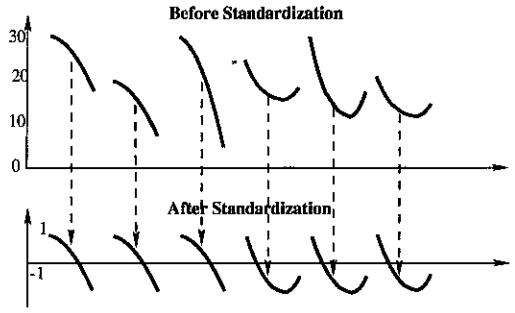
Figure 6(a) illustrates the transformation. It shows that two patterns of the same shape will be transformed to identical patterns in term of the resulting transformed values, regardless of the absolute level (μ) of the input pattern and the magnitude of the variations within the input patterns (captured by c). Figures 6(b) and 6(c) depict the desired outputs before and after the transformation. Clearly, the nonstationarity, especially the long term trend, is removed. An undesirable feature of the method, however, is that it over-amplifies the flat regions of the time series and may cause instability in predictions in these regions.

2.3. *Denoising the high-frequency signals*

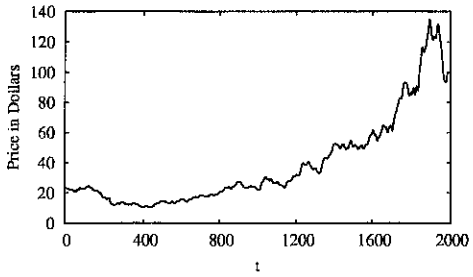
As is shown in Fig. 4, the high-pass channels contain noise that makes predictions and generalization difficult. In this section, we propose a channel-specific denoising strategy for removing noise in each channel. Since denoising by causal filters incurs delays in the filtered signals, we like to find an appropriate filter for each channel in order to remove its noise and to incur the minimum delay. No denoising will be needed in the low-pass channel, because it is already noise-free.

Figure 7 illustrates the filter coefficients for each channel in the 3-channel wavelet decomposition shown in Fig. 4. It also shows the coefficients of the corresponding channel-specific causal filters for denoising Channels H and LH.

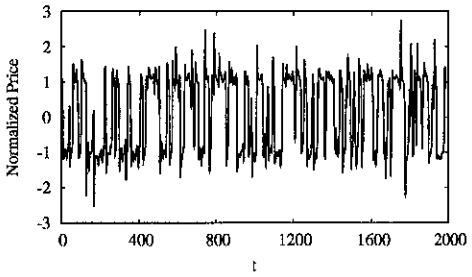
Our extensive experimental results (not shown[16]) show that the best prediction quality is obtained by a three-channel wavelet decomposition with a B_3-spline



(a) Standardization of each pattern of a nonsationary time series.



(b) Original low-pass signals.



(c) Transformed time series.

Fig. 6. Transforming the nonstationary series of IBM's low-pass signals into a stationary chaotic time series by performing standardization on its input patterns.

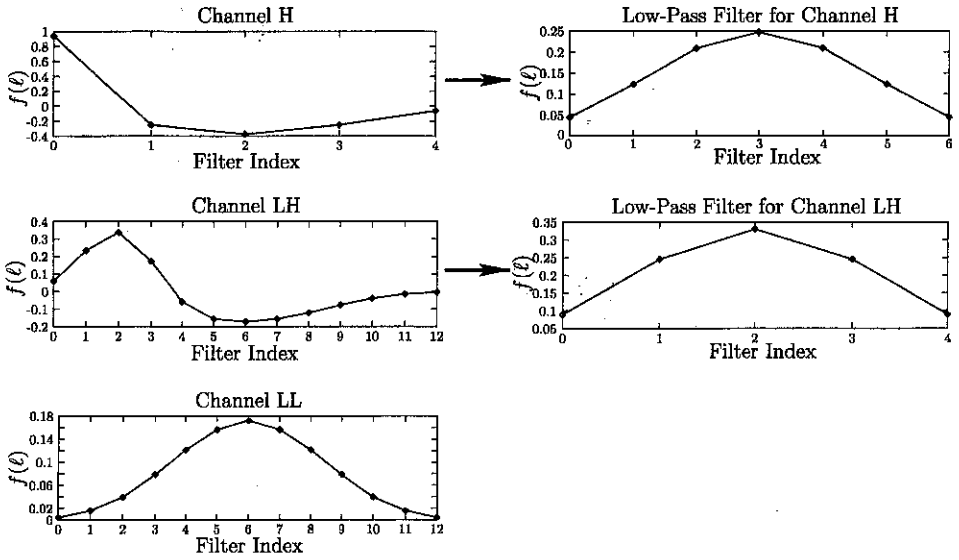
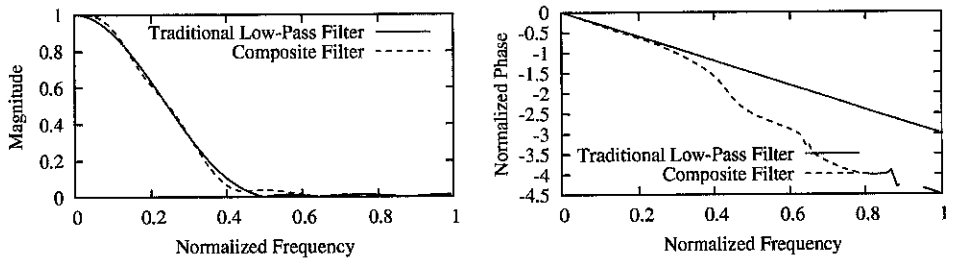


Fig. 7. Wavelet decomposition and channel-specific low-pass filtering for H and LH. The left panels plot the filter coefficients of each wavelet channel, whereas the right panels plot the coefficients of the causal filters that are applied to the two high-frequency channels. The horizontal axis represents the indexes for each filter, and the vertical axis indicates the values of the coefficients.



(a) Magnitude response. (b) Phase response.

Fig. 8. Frequency response of the composite filter.

mother filter, and channel-specific causal filters of seven taps with 0.15 cutoff for Channel H and of five taps with 0.15 cutoff for Channel LH (see Fig. 7). As a result, we use this configuration in our experimental results in Sec. 4.

After each channel is denoised, the combined signals can be constructed by summing the preprocessed signals from each channel into a single stream and by deriving the coefficients of the equivalent composite filter.[16] Using these filter coefficients, we then find the frequency response of the composite filter by FFT.

Figure 8 plots the frequency response of the composite filter when compared to a traditional low-pass filter with seven taps and 0.15 cutoff (7/0.15 filter). Figure 8(b) shows some phase distortions in the high-frequency range that are introduced by the nonlinearity of the composite filter. These distortions are not critical because the magnitude of the frequency response is small in the high-frequency range (Fig. 4).

The main advantage of a multi-channel wavelet decomposition with channel-specific low-pass filtering is that it allows specialized predictors to be designed to tailor to the property of each channel and to achieve the best prediction accuracy for each. In particular, the low-pass channel with the largest magnitude is already noise free and does not require further low-pass filtering. Hence, it can be predicted very accurately into the future. On the other hand, the two high-frequency channels require further low-pass filtering for removing undesirable noise. The prediction errors in these channels, however, are less significant because the magnitudes of the signals in these channels are much smaller when compared to those in the low-pass channel. We demonstrate in Sec. 4 the improvements of channel-specific filtering over the traditional approach of a single low-pass filter in terms of hit ratios and annual returns.

2.4. *Compensating for edge effects*

To handle the edge effects of a filtered time series, existing schemes extend the raw data $R(t)$ into the future in order to obtain low-pass data up to the current time t_0. Figure 9 illustrates four common techniques: wrap-around, mirror extension,[10] flat extension, and zero-padding.[15]

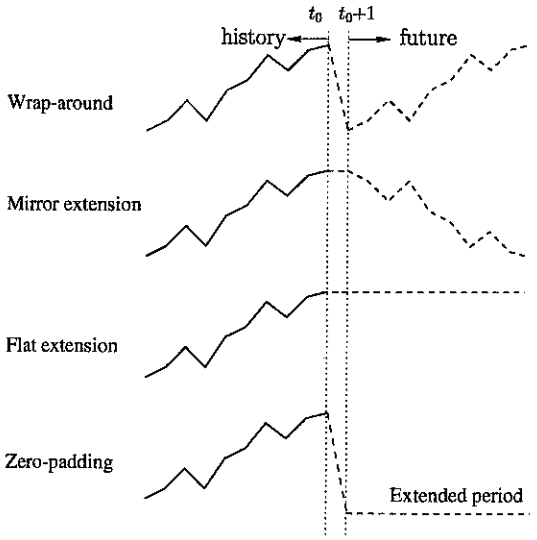


Fig. 9. Four approaches for handling the edge effect by extending the raw data beyond t_0 before denoising. Solid lines represent the raw data up to t_0, and dashed lines illustrate the extensions to the raw data made from $t_0 + 1$ and beyond.

For example, consider a 21-tap causal low-pass filter. Low-pass data is available up to t_0 but is delayed by 10 days, because a 21-tap causal filter incurs a 10-day delay in its filtered data. To overcome the delay, we will need raw data $R(t_0 + 1)$ to $R(t_0+10)$ in order to generate $S(t_0+1)$ to $S(t_0+10)$. To obtain this missing data, flat extensions (resp., mirror extensions) assume that future raw data $R(t_0+h) = R(t_0)$ (resp., $R(t_0 + h) = R(t_0 - h + 1)$) for all $h = 1, \ldots, 10$ before the 21-tap low-pass filter is applied to obtain $S(t_0 + 1)$ to $S(t_0 + 10)$.

Figure 9 clearly shows that, when a trend is present in the raw data, wrap-around and zero-padding do not work well because they have abrupt transitions at t_0. In contrast, mirror and flat extensions work better in some part of the extended period. Figure 10 shows that both have small average errors for the first seven estimated low-pass points in the extended period with respect to the true low-pass data, and flat extensions perform slightly better. Other results 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 extended 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 extended period between the raw and the fitted smoothed data. Figure 10 shows that the average errors achieved by polynomial fits of degree one are generally larger than flat extensions, while fits of higher degrees give worse errors (not shown). It should be noted that the errors are particularly large at the beginning of the extended period because the fitted curve is not constrained to have small errors at the transition point.

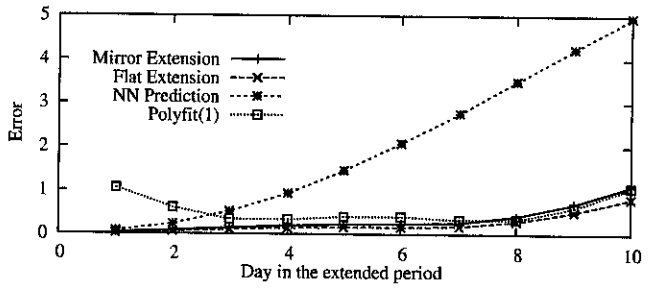


Fig. 10. Average errors of IBM's closing prices with respect to the true 21-tap low-pass data between April 1, 1997 and March 31, 2002, on four approaches for handling edge effects.

Yet another method studied is to train an ANN using the low-pass data available in order to predict the low-pass data in the extended period. Assuming an extended period of size m, we used a constrained ANN formulation but without the constraints on cross-validation proposed in Sec. 3, trained the ANN to perform one-step predictions using patterns up to $t_0 - m$, and applied the ANN to perform iterative predictions on the data in the rest of the extended period. The results in Fig. 10 clearly show that such an approach performs poorly.

Among all the methods tested, Fig. 10 shows that flat extensions achieve the smallest average errors for the first seven days of the extended period, but have considerably larger errors in the last three days. Similar observations have been found in the filtered signals of different delays. Therefore, in our experiments, we use flat extensions to generate new training patterns in the extended period.

3. Constrained Formulation and ANN Learning Algorithm

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 due to delays in the filtered sequence. Since the problem as formulated cannot be solved by conventional ANN learning methods, we apply a constrained formulation and learning strategy we have developed earlier for predicting chaotic time series with similar characteristics.[17] In our formulation in this paper, we also propose new constraints on cross validations.

In our formulation, we express the learning objective to be the minimization of the sum of squared one-step prediction errors over a window W of training patterns in the RFIR in Fig. 3. We use the one-step error instead of the more complex iterative prediction error over a horizon because it allows gradients to be easily computed by back-propagation. We handle the complex iterative prediction errors by formulating them as constraints in cross validations. We further introduce a constraint on the output error of each training pattern as follows:

$$p_i^p(w) = (o(i) - d(i))^2 \leq \tau_i^p, \quad 1 \leq i \leq W, \tag{7}$$

where $o(i)$ and $d(i)$ are, respectively, the ith actual and desired (target) outputs, and τ_i^p is the error tolerance.

Next, we introduce new constraints to limit the errors from multiple validation sets in cross validations. In traditional approaches, cross validations involve the computation of a performance measure on a set of training patterns after learning is completed. Since the measure used in cross validations is the same as that in learning, patterns used in cross validations must be different from those used in learning. Our 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.

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 known training patterns. Here, we define a *validation set* to be a collection of $L = m + h$ training patterns, where m is the delay because causal filtering 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, 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 extended period and substantial errors are incurred on patterns beyond the extended 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 is done in our previous work,[17] 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 the hit ratio defined in Sec. 1, there are $2h$ additional constraints:

$$p_e^v(w) \leq \tau_e^v, \quad p_e^r(w) \leq \tau_e^r \quad \text{for } 1 \leq e \leq h, \tag{8}$$

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

Putting all the constraints together, we have:

$$\begin{aligned} \min_w & \sum_{i=1}^{W} \max\left\{(o(i) - d(i))^2 - \tau, 0\right\} \\ \text{s.t. } & p_i^p(w) \leq \tau_i^p, \quad 1 \leq i \leq W, \\ & p_e^v(w) \leq \tau_e^v, \quad 1 \leq e \leq h, \\ & p_e^r(w) \leq \tau_e^r. \end{aligned} \tag{9}$$

Since Eq. (9) is a constrained nonlinear programming problem (NLP) with non-differentiable functions, it cannot be solved by the traditional back-propagation algorithm that require the differentiability of functions. To address this issue, we apply a violation-guided back-propagation algorithm (VGBP) we have

developed[7,16,17] for solving this constrained problem. VGBP works on the penalty function transformed from Eq. (9) and searches for extended saddle points.[18] It does this by gradient descents in the original weight space and ascents in the penalty space, using an approximate gradient of the objective function found by back-propagation and according to the violation of each constraint.

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 harder to satisfy, while achieving smaller violations at convergence. By using loose constraints in the beginning and by gradually tightening the constraints, learning converges faster with tighter tolerances.

4. Experimental Results

In this section, we evaluate the prediction quality of CNN-CSP, our proposed constrained ANN trained by VGBP, when applied on data filtered by channel-specific low-pass processing. In predicting Channel LL, we use a recurrent FIR ANN with a $1:20:1$ structure (namely, 1 input node, 20 hidden nodes, and 1 output node) and a $15:0:0$ filter structure (namely, a 15-tap filter coming into the input node, and no FIR filter for connections between the input and the hidden layers and between the hidden and the output layers). The activation function for the hidden (resp., output) nodes is the $\tanh(x)$ (resp., linear) function. The network uses the 200 most recent prices for learning. With respect to Channels LH and H, we combine them into a single channel and do not carry out predictions for each individually. Each of these channels do not give meaningful hit ratios for the predictions because their filtered low/high data crosses each other. To predict this combined channel, we use a recurrent FIR ANN with a $1:12:1$ structure, a $15:0:0$ filter structure, similar activation functions as before, and a window of the 120 most recent patterns.

We compare the performance of CNN-CSP against two predictors: auto-regression (AR) of order n using the *TISEAN* implementation,[19] and CNN-LP, a constrained ANN trained by VGBP on a single low-pass time series. For CNN-LP, after extensive experimentation,[16] we have chosen a low-pass filter with seven taps and a cut-off frequency of 0.15. We have tested our predictors using the closing prices of IBM (symbol IBM), Citigroup (symbol C) and Mentor Graphics (symbol MNTR) from November 1992 to February 2003. In our experiments, we do not combine the prediction results of multiple stocks to form a multi-stock portfolio.

To evaluate the cumulative returns of these predictors, we compare them using two trading strategies based on an initial portfolio of \$1 million. The first strategy tested is the buy-and-hold strategy that makes a purchase at the beginning of the period and holds the position during the period without making any transaction. We also develop a simple trading strategy that works as follows. We assume that the daily low/high prices are known a few minutes before the market closes at Day t_0. Based on the predicted price for $t_0 + 1$ before the market closes, if this price is

higher than the daily high price for t_0, then we buy the stock if cash is available; if the predicted price for t_0+1 is lower than the daily low price, then we sell the stock if there are shares in the portfolio. Next, right at the moment when the market opens at t_0+1, we immediately buy (resp., sell) as many shares as possible if the predicted price for t_0+1 is higher (resp., lower) than the opening price.

Since, our simple trading strategy is applied on a filtered time series with delays and does not manage risks, it is not robust to unpredictable changes in the market. To address the effects of risks, we have developed a simple heuristic post-processing strategy. Its main idea is to detect if the previous prediction $P_{t_0-1}(t_0)$ for t_0 is outside the actual daily low/high price range at t_0. If it is outside the range, then a correction is applied on the current day's prediction $P_{t_0}(t_0+1) = 0.25\,(R_\ell(t_0) + R_h(t_0)) + 0.5R_c(t_0)$, where $R_\ell(t)$, $R_h(t)$, and $R_c(t)$ are, respectively, the daily low, high, and closing prices at t. Otherwise, no correction is performed.

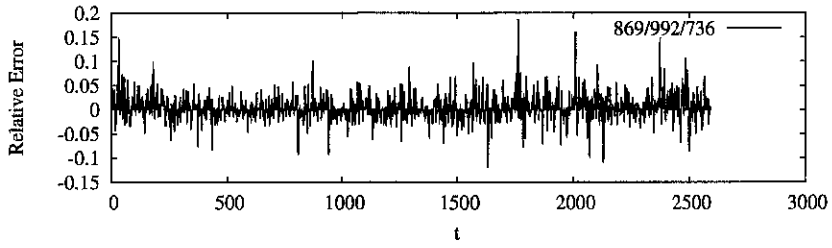
Figure 11 plots the prediction errors in terms of the number of predictions with zero/positive/negative errors for IBM. Here, a zero (resp., positive and negative) error means that the prediction is within the daily low/high range (resp., above the daily high and below the daily low). Due to space limitation, we do not show the results for other time series.[16] Our results show that AR has the worst performance, whereas CNN-CSP consistently outperforms CNN-LP. Since the only difference between CNN-LP and CNN-CSP is in their channel-specific preprocessing, we conclude that such preprocessing is effective for reducing range errors.

Table 1 summarizes the hit ratios with respect to the daily low/high prices, along with their average magnitudes of errors. The results show that CNN-CSP works better because it has a high hit ratio on zero errors and a low magnitude in its average positive/negative prediction errors.

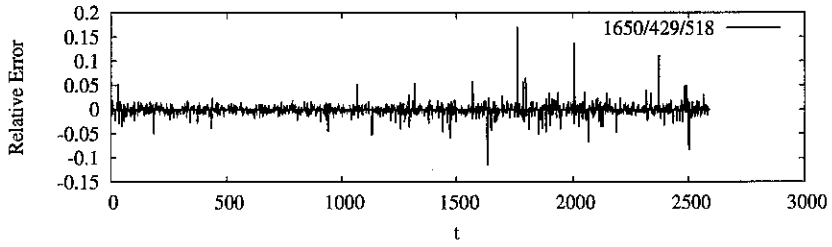
Figure 12 plots the evolution of the portfolios when using our simple trading strategy and the predicted data from the three predictors. Table 2 also shows the annual return of each portfolio for ten stocks. It confirms one observation in the literature[20] that complex predictors (such as AR and CNN-LP) do not always outperform the simple buy-and-hold strategy in the long term.

It is interesting that the portfolio performance of AR is not much better than buy-and-hold: five out of ten portfolios based on AR are better and the rest are worse. There is also little difference between the two strategies in terms of their average annual returns. For the strategy based on CNN-LP, it beats buy-and-hold in only three out of the ten portfolios. On the other hand, the strategy based on CNN-LP has higher average annual returns than buy-and-hold. It significantly under-performs buy-and-hold in only three portfolios (C, PSS, and YHOO) but performs much better in the portfolios for MNTR and PFGI. When compared with AR, CNN-LP performs better in six out of the ten portfolios and has higher average annual returns. Based on the above observations, we conclude that AR and CNN-LP do not perform better than buy-and-hold.

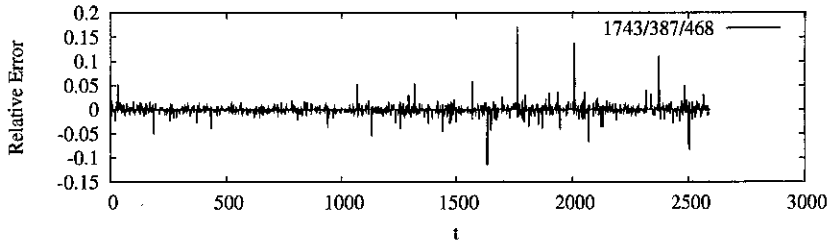
On the other hand, CNN-CSP is almost consistently better than CNN-LP (in eight of the ten portfolios) and has significantly higher average annual returns than



(a) Auto-regressive Model (AR).



(b) Predictor CNN-LP.

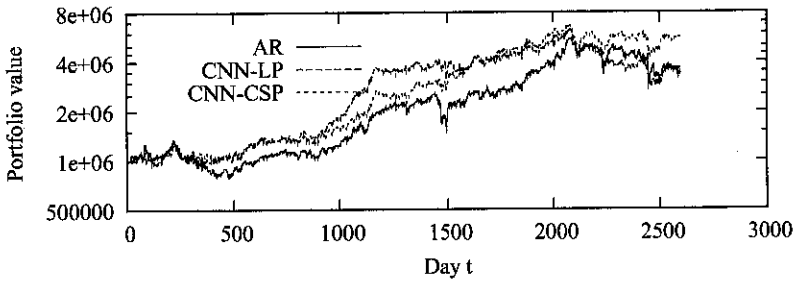


(c) Predictor CNN-CSP.

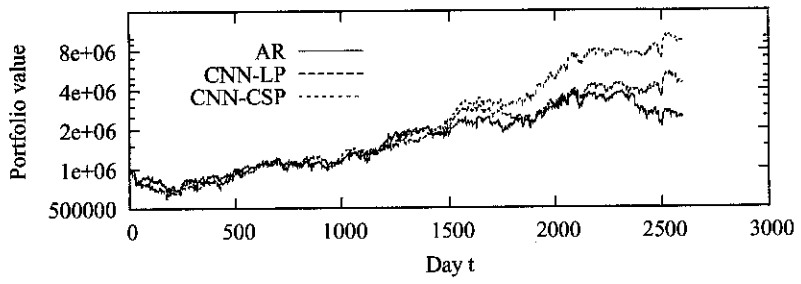
Fig. 11. Zero/positive/negative errors with respect to the daily low/high stock prices for IBM from November 4, 1992 to February 28, 2003.

Table 1. Hit ratios and the corresponding average errors for the three predictors.

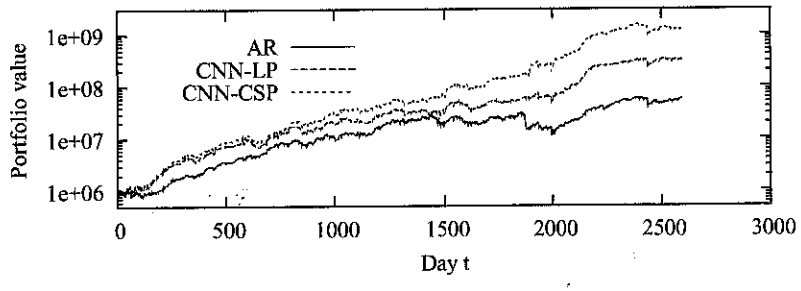
Stock	Predictor	α_0	(α_+, Avg. Error)	(α_-, Avg. Error)
C	AR	0.324	(0.385, 0.023)	(0.292, −0.019)
	CNN-LP	0.598	(0.178, 0.008)	(0.224, −0.009)
	CNN-CSP	0.632	(0.161, 0.007)	(0.206, −0.008)
IBM	AR	0.335	(0.382, 0.022)	(0.283, −0.017)
	CNN-LP	0.635	(0.165, 0.009)	(0.200, −0.009)
	CNN-CSP	0.671	(0.149, 0.008)	(0.180, −0.008)
MNTR	AR	0.398	(0.379, 0.032)	(0.223, −0.022)
	CNN-LP	0.747	(0.128, 0.013)	(0.125, −0.011)
	CNN-CSP	0.801	(0.100, 0.011)	(0.099, −0.008)



(a) Citigroup.



(b) IBM.



(c) Mentor Graphics.

Fig. 12. Portfolio value grows with time for trading Citigroup, IBM, and Mentor Graphics stocks based on Predictors AR, CNN-LP, and CNN-CSP. The stock prices at the beginning and the ending days of the period are, respectively, \$2.880 and \$33.390 (Citigroup), \$16.895 and \$77.735 (IBM), and \$4.200 and \$17.375 (MNTR).

buy-and-hold, AR, and CNN-LP, although it performs considerably worse than buy-and-hold in PSS. Since the only difference between CNN-CSP and CNN-LP is in their preprocessing, we conclude that channel-specific preprocessing is the key to improved portfolio performance. Further, CNN-CSP outperforms AR and buy-and-hold in six out of the ten portfolios and is robust even when it underperforms. Overall, the average annual return over the ten portfolios for CNN-CSP is significantly better than any of the other three strategies.

Table 3 shows the annual return by year for CNN-CSP. The large variations in annual returns from one year to another is typical when there is no risk control.

Table 2. Annual percentage returns M(year) of portfolios based on the simple trading strategy using one of the three predictors and the buy-and-hold strategy. All returns are based on portfolio values without leverage. The highest annual return achieved for each stock is underlined.

Stock	Buy-and-Hold(%)	AR(%)	CNN-LP(%)	CNN-CSP(%)
AMR	−22.07	−15.53	−23.77	−23.45
C	26.77	12.60	13.71	18.61
GE	15.76	10.27	14.88	15.13
IBM	15.92	9.11	15.68	24.16
MNTR	14.74	47.87	74.69	99.05
NYT	13.94	21.12	10.92	15.60
PFGI	11.02	29.06	55.92	53.16
PSS	1.23	−3.62	−11.83	−13.45
XOM	8.19	18.87	10.61	12.10
YHOO	47.54	12.54	31.92	48.61
Average	13.30	14.23	19.27	24.95

Table 3. Annual percentage returns M(year) for portfolios based on CNN-CSP.

Year	AMR	C	GE	IBM	MNTR	NYT	PFGI	PSS	XOM	YHOO
1993	−9.99	−0.10	19.83	4.32	288.02	11.39			13.43	
1994	−25.76	5.11	12.15	15.66	143.51	27.40	127.45		7.64	
1995	15.62	17.59	7.14	15.53	86.46	83.21	96.49		19.84	
1996	2.38	34.73	9.96	28.80	114.51	10.93	139.47		12.58	
1997	−3.83	44.95	5.11	31.50	41.71	38.20	114.85		57.48	
1998	−10.89	25.86	23.16	69.39	98.96	−6.00	34.68	−49.59	17.07	138.05
1999	−13.39	49.25	−17.84	3.03	36.61	21.80	42.36	−8.00	−2.64	118.44
2000	−41.39	27.25	−7.44	85.67	162.32	−6.57	−22.94	14.71	7.55	−70.24
2001	−56.30	−5.46	−10.79	35.60	201.42	14.06	18.15	−11.53	−4.74	23.03
2002	−43.59	−8.83	41.93	20.56	1.85	−10.84	4.16	−12.66	13.12	70.57

Also, Years 2000 and 2001 are the most difficult years for the long-only portfolios based on our simple naive trading strategy and CNN-CSP. The global economic environment in those years are very tough for long-only portfolio managers because of the burst of the technology bubble and the September 11 attack.

Finally, we analyze the reasons for the failure of our strategies in some periods. Firstly, our predictors may fail when some drastic price changes either were never learned or occurred very infrequently in the historical information. For example, based the daily closing prices for AMR that increased from \$51.50 on March 7, 2000 to \$60.63 on March 15, 2000, our predictors predicted that the stock price of AMR would continue its uptrend. However, on March 16, 2000, the closing price dropped to \$28.00, and all the portfolios suffered more than 50% loss in that single day. Similarly, before July 1, 2002, the price for PSS was on an up-trend, but its price dropped over 25% between July 1, 2002, and July 23, 2002. Our predictors had trouble in predicting such big trend switches and ended losing money.

Second, our simple trading strategies only maintain long positions and do not perform well when prices are continuously dropping. For instance, YHOO experienced the typical technology stock bubble burst from March 2000 to the middle of 2002, and all our long-only portfolios based on the three predictors experience continuous loss. Similarly, the stock price for PSS dropped continuously from July 1998 to October 1998, and all three long-only portfolios can only lose money.

Third, our simple trading strategies do not consider important events that may affect stock prices. In reality, stock prices are greatly influenced by many factors, such as recent trading volumes, macro economic factors, industry/sector performance, analyst's upgrade/downgrade recommendations, scandals, and events like the September 11 terrorist attack.[21] In this case, historical stock prices are weakly correlated to future prices.

5. Conclusions

In this paper, we have studied the prediction of stock prices modeled as a univariate time series. We have proposed a number of pre-processing techniques for improving its predictability. Specifically, we have applied wavelet decomposition and channel-specific low-pass filtering for removing undesirable high-frequency noise. We have presented transformation techniques for improving the stationarity of the low-frequency component and have evaluated methods for compensating edge effects due to denoising. By using a constrained formulation in ANN learning, we have the flexibility of using multiple validation sets and of adding new constraints on cross-validation errors. Finally, we present some experimental results on ten stock benchmarks using three predictors: Autoregressive model, ANN model with traditional low-pass preprocessing, and ANN model with channel-specific preprocessing.

Our results show that channel-specific preprocessing is superior to traditional low-pass filtering when used in our ANN predictors. When compared to the autoregressive and buy-and-hold strategies, the results show that our approach has better forecasting power and are consistent across a variety of stocks over a period of ten years. Our analysis also shows that macro-economic and social factors may need to be incorporated in learning in order to achieve better predictions.

Acknowledgment

Research supported by National Science Foundation Grant IIS 03-12084.

References

1. C. Chatfield, *Time-Series Forecasting* (Chapman & Hall/CRC, Boca Raton, Florida, 2001).
2. G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, 2nd ed. (Holden-Day, San Francisco, 1976).
3. M. Aoki, *State Space Modeling of Time Series* (Springer-Verlag, Berlin, 1987).

4. D. F. Nicholls and A. R. Pagan, Varying coefficient regression, in *Handbook of Statistics*, eds. E. J. Hannan, P. R. Krishnaiah and M. M. Rao, (North-Holland, Amsterdam, 1985), pp. 413–449.
5. H. Tong, *Nonlinear Time Series: A Dynamical System Approach* (Oxford University Press, Oxford, 1990.
6. S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. (Prentice Hall, NJ, 1999).
7. B. W. Wah and M.-L. Qian, Violation guided neural-network learning for constrained formulations in time-series predictions, *Int'l J. Computational Intelligence and Applications* **1**(4) (2001) 383–398.
8. J. L. Elman, Finding structure in time, *Cognitive Science* **14** (1990) 179–211.
9. E. A. Wan, Finite impulse response neural networks with applications in time series prediction., Ph.D. Thesis, Standford University (1993).
10. G. Zheng, J. L. Starck, J. G. Campbell and F. Murtagh, Multiscale transforms for filtering financial data streams, *J. Computational Intelligence in Finance* **7** (1999) 18–35.
11. T. Hellstrom and K. Holmstrom, *Predicting the Stock Market*, Technical Report Series IMa-TOM-1997-07, Malardalen University, Vasteras, Sweden (1997).
12. R. W. Hamming, *Digital Filters* (Prentice-Hall, Englewood Cliffs, NJ, 1989).
13. R. D. Edwards and J. Magee, *Technical Analysis of Stock Trends*, 5th ed. (John Magee, Springfield, MA, 1966).
14. B.-L. Zhang, R. Coggins, M. A. Jabri, D. Dersch and B. Flower, Multiresolution forecasting for future trading using wavelet decompositions, *IEEE Trans. on Neural Networks* **12**(7) (2001) 766–775.
15. T. Masters, *Neural, Novel and Hybrid Algorithms for Time Series Prediction* (John Wiley & Sons, Inc., NY, 1995).
16. M. L. Qian, Neural network learning for time-series predictions using constrained formulations, Ph.D. Thesis, Department of Computer Science Report No. 2437, University of Illinois, Urbana, IL (2005).
17. 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)* (2001), pp. 771–776.
18. B. Wah and Y. X. Chen, Constraint partitioning in penalty formulations for solving temporal planning problems, *Artificial Intelligence* **170**(3) (2006) 187–231.
19. R. Hegger and T. Schreiber, *The TISEAN Software Package*, http://www.mpipks-dresden.mpg.de/tisean (2002).
20. S. Makridakis and M. Hibon, The M3-Competition: Results, conclusions and implications, *Int. J. Forecasting* **16** (2000) 451–476.
21. R. Haugen, *New Finance: The Case Against Efficient Markets* (Prentice Hall, Englewood Cliffs, NJ, 1998).