International Journal of Neural Systems, Vol. 8, Nos. 5 & 6 (October/December, 1997) 517–534
© World Scientific Publishing Company

ADAPTIVE RIVAL PENALIZED COMPETITIVE LEARNING AND COMBINED LINEAR PREDICTOR MODEL FOR FINANCIAL FORECAST AND INVESTMENT

YIU-MING CHEUNG, WAI-MAN LEUNG and LEI XU
Department of Computer Science and Engineering,
The Chinese University of Hong Kong, Shatin, Hong Kong

Received 25 July 1996
Revised 18 September 1997
Accepted 4 November 1997

We propose a prediction model called *Rival Penalized Competitive Learning* (RPCL) and *Combined Linear Predictor* method (CLP), which involves a set of local linear predictors such that a prediction is made by the combination of some activated predictors through a gating network (Xu *et al.*, 1994). Furthermore, we present its improved variant named Adaptive RPCL-CLP that includes an adaptive learning mechanism as well as a data pre-and-post processing scheme. We compare them with some existing models by demonstrating their performance on two real-world financial time series — a China stock price and an exchange-rate series of US Dollar (USD) versus Deutschmark (DEM). Experiments have shown that Adaptive RPCL-CLP not only outperforms the other approaches with the smallest prediction error and training costs, but also brings in considerable high profits in the trading simulation of foreign exchange market.

1. Introduction

The classical statistical models such as MA and ARMA cannot model a financial series satisfactorily due to its non-stationarity and high-frequency fluctuation. In the past, there has been considerable interest in exploring the applications of artificial neural networks (ANNs) to financial series prediction.[2,4–7,10,11,14] An advantage of the ANN based method is that it does not require us to predefine a particular nonlinear model. Alternatively, these methods try to establish a functional relationship between the input patterns and the outputs directly through an elaborate iterative learning scheme without consideration of the specific properties of financial data. But it usually results in large computing costs.

One feasible way is to use a set of linear systems to model a nonlinear environment approximately.[3] The basic idea is that we separate a global nonlinear environment into some small regions, in each of which we regard it as a linear one modeled by a linear system. In this paper, we propose such a model called *Rival Penalized Competitive Learning*[18] and *Combined Linear Predictor* method (RPCL-CLP), whose implementation consists of two stages — training stage followed by prediction stage. In the training stage, RPCL-CLP constructs a set of Local Linear Predictors (LLP) by regressing those input-and-output pairs within each cluster respectively. In the prediction stage, a prediction is made by combining some activated Local Linear Predictors through a gating network (G.N.).[17]

Moreover, we propose its adaptive variant named Adaptive RPCL-CLP, where a data pre-and-post processing scheme is involved to eliminate some common-factor effects. In the training stage, Adaptive RPCL-CLP is learned in the same way as RPCL-CLP. However, in the prediction stage, the model parameters are updated adaptively upon each prediction made instead of being fixed in the original RPCL-CLP. Furthermore, Adaptive RPCL-CLP also combines the Random Walk model, which is ideal to model a market with perfect efficiency,[1] with the prediction of triggered Local Linear Predictor via winner-take-all through the gating network.[15]

We compare our proposed models with two popular time-delay recurrent networks: *Elman Net*[2] and *Jordan Net*[5–8] as well as the classical MA(q) and Random Walk models by demonstrating their performance on two real-world time series — a China stock price and an exchange-rate series of US Dollar (USD) versus Deutschmark (DEM). Also, we design a trading model with three various risk-level strategies for the investment simulation in the foreign exchange market. Experimental results have shown that Adaptive RPCL-CLP not only outperforms the other models with the smallest prediction error and training costs, but also brings in considerable high profits under the trading model.

This paper is organized as follows: Section 2 introduces the extraction of the input patterns and desired outputs. Sections 3 and 4 describe the architecture and implementations of RPCL-CLP and Adaptive RPCL-CLP respectively. A trading model with three various risk-level strategies is described in Sec. 5 and computer experiments are given in Sec. 6. We draw a conclusion in Sec. 7.

2. Extraction of Input Patterns and Outputs

We separate a series $\{z_t\}_{t=1}^{N+S}$ into two parts: $\{z_t\}_{t=1}^{N+1}$ (training series) and $\{z_t\}_{t=N+2}^{N+S}$ (testing series). The training set $\{(Z(t),\ U(t))\}_{t=1}^{N+1-d}$ can be extracted from the training series $\{z_t\}_{t=1}^{N+1}$ by successively shifting a sliding window as shown in Fig. 1, where we denote input pattern $Z(t) = [z_t, z_{t+1}, \ldots, z_{t+d-1}, z_{t+d}]$ and desired output $U(t) = z_{t+d+1}$ at time t respectively. The testing set $\{Z(t)\}_{t=N-d+2}^{N+S-d}$ can be obtained in the same way, where the desired targets are unknown.

3. RPCL-CLP Model

3.1. *RPCL-CLP architecture*

As shown in Fig. 2, the RPCL-CLP architecture has four layers: Input Layer, Cluster Layer, Combination Layer and Output Layer, where the inputs $Z(t)'$s and model outputs are embedded in the

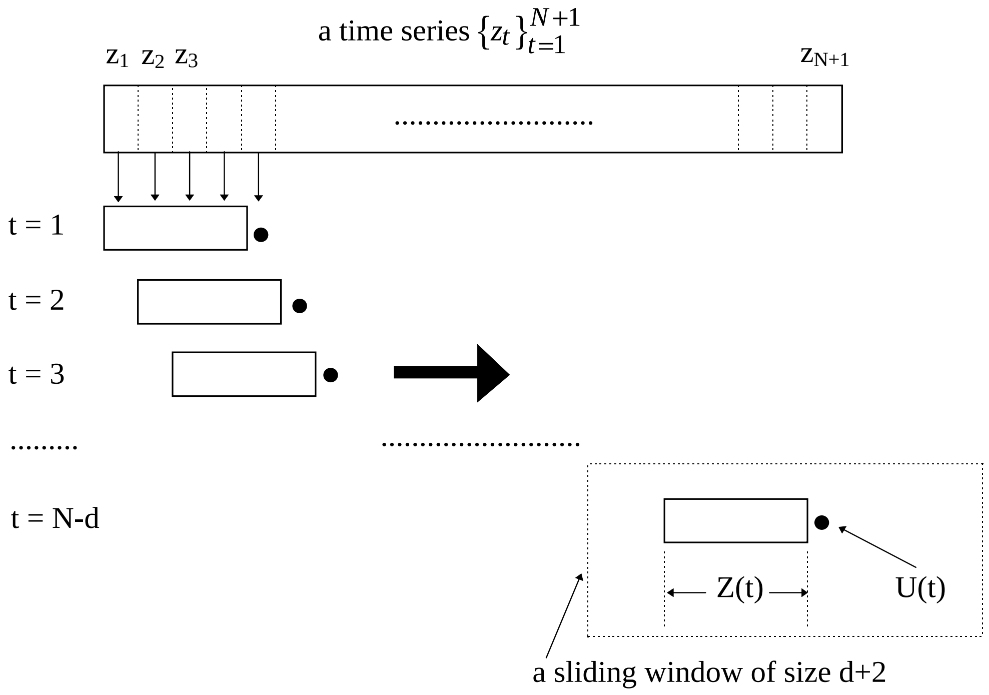

a time series $\{z_t\}_{t=1}^{N+1}$
z1 z2 z3
zN+1
t = 1
t = 2
t = 3
t = N-d
Z(t)
U(t)
a sliding window of size d+2

Fig. 1. A sliding window scans through a training series $\{z_t\}_{t=1}^{N+1}$ to extract a set of input-output pairs $\{(Z(t),\ U(t)\}_{t=1}^{N-d}$, where the dimension of input space is set at $d+1$.

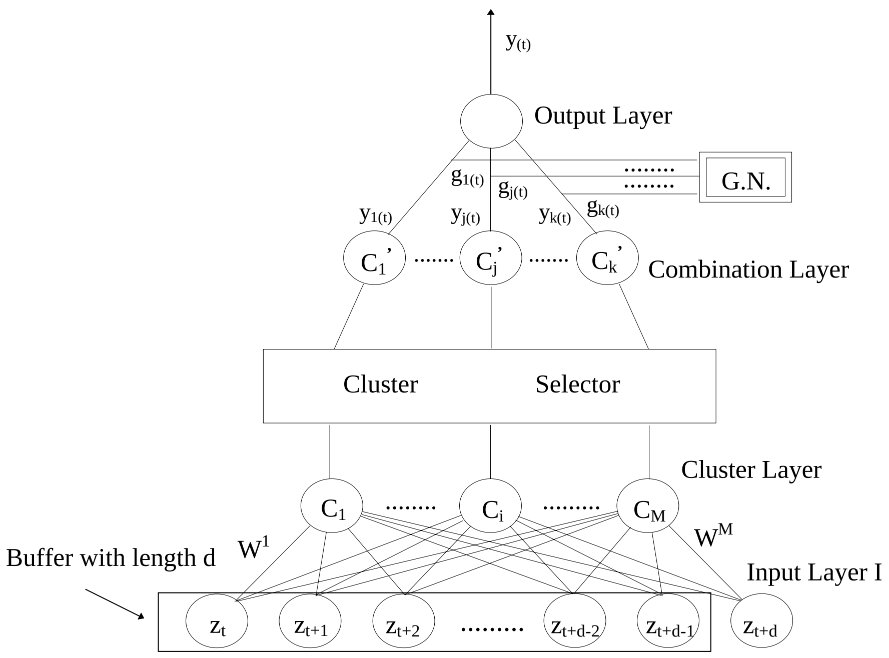


Fig. 2. Architecture of RPCL-CLP model, where we use a buffer storing previous d input data points which form an input pattern together with the current input data point z_{t+d}.

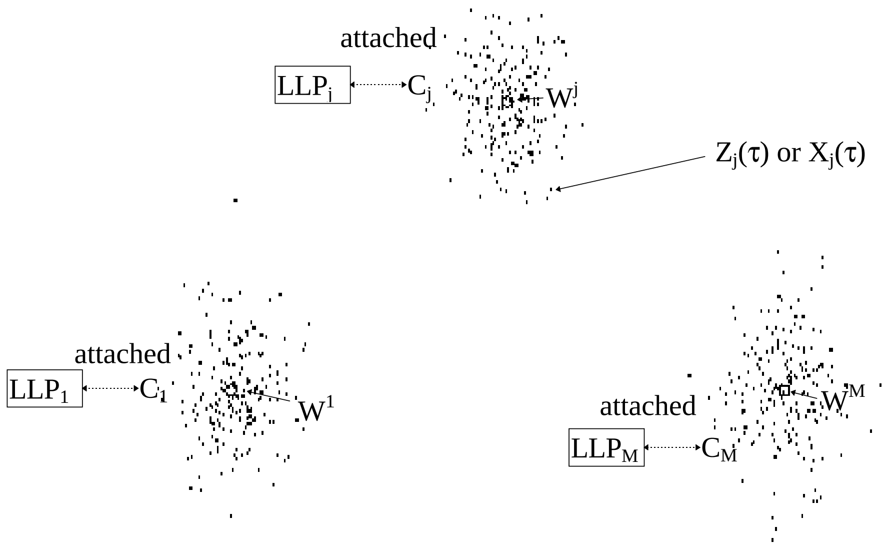


Fig. 3. Internal structure of the cluster nodes. Each node C_j includes a cluster centroid W^j and a group of input patterns $Z_j(\tau)$ (in RPCL-CLP) or $X_j(\tau)$ (in Adaptive RPCL-CLP), attached with a local linear predictor LLP_j learned by least square method.

Input Layer and Output Layer respectively. The Cluster Layer consists of a set of cluster nodes, each of which includes a cluster centroid and a group of input patterns assigned by the unsupervised learning approach in the training stage. Also, as shown in Fig. 3 each cluster node is attached with a Local Linear Predictors (LLP) by regressing those input-output pairs within the cluster node. The

Combination Layer consists of those cluster nodes selected by the Cluster Selector, where the attached LLP's are activated and their outputs $y_{j(t)}$'s ($j = 1, \ldots, k$) are combined by the gating network (G.N.)[17] to form $y(t)$, i.e. the estimation of $U(t)$.

3.2. *Training stage of RPCL-CLP*

The training of RPCL-CLP includes three steps: (I) Creating Cluster Layer dynamically. (II) Building up Local Linear Predictors (LLP) for each cluster node. (III) Training the gating network.

3.2.1. *Creation of cluster layer*

The procedure of building cluster layer consists of two phases. In Phase 1, we set up a preliminary Cluster Layer with K cluster nodes by Incremental Clustering algorithm (IC), where we use a heuristic algorithm to automatically control the ε (a threshold value for creating a new cluster node). However, as pointed out in Ref. 18, some of these resulted cluster nodes centroids may not locate at the center of the corresponding clusters. They may be either at some boundary points between different clusters or at points biased from some cluster centers. These centroids may draw quite a large portion of samples from correct clusters to form some disturbing groups that lead to an incorrect clustering results. Hence, in Phase 2 we apply Rival Penalized Competitive Learning (RPCL)[18] to refine these initial cluster nodes. The basic idea is that for each input pattern not only the centroid W^c which wins the competition is modified to adapt to the input, but also the centroid W^r of its rival is de-learned by a learning rate smaller than that used by W^c.

Specifically, the detailed algorithm is described as follows:

Phase 1

We create an initial Cluster Layer by a modified Incremental Clustering algorithm:

Step 1. Given a time series $\{z_t\}_{t=1}^{N+1}$, we calculate the covariance matrix Σ of the input patterns $\{Z(t)\}_{t=1}^{N+1-d}$ first.

Step 2. Let ε be the square root of the smallest element in the main diagonal of Σ as a threshold for creating a new cluster node.

Step 3. Let

$$W^1 = Z(1) \tag{1}$$

where W^1 is the centroid of the first cluster node C_1.

Step 4. Assume that the jth cluster nodes have been created dynamically and each C_i has a centroid W^i. Given a new input pattern $Z(t)$, we calculate the distance between it and each cluster's centroid by

$$d_i = \|Z(t) - W^i\|, \text{ where } i = 1, 2, \ldots, j. \tag{2}$$

to find out one with the minimum distance d_m. If $d_m \le \varepsilon$, we assign $Z(t)$ to this cluster node C_m and update its centroid W^m by

$$n_m^{\text{new}} = n_m^{\text{old}} + 1 \tag{3}$$

$$W_{\text{new}}^m = W_{\text{old}}^m + \frac{1}{n_m^{\text{new}}}(Z(t) - W_{\text{old}}^m) \tag{4}$$

where n_m is the number of input patterns assigned to node C_m.

Otherwise, we create a new cluster node C_{j+1}. Let

$$W^{j+1} = Z(t) \tag{5}$$

This step is iterated until all data points have been assigned.

Phase 2

We adjust these initial cluster nodes centroids by RPCL algorithm as follows:

Step 1. For each input pattern $Z(t)$, let

$$u_{i(t)} = \begin{cases} 1, & \text{if } i = c \text{ such that } \gamma_c\|Z(t) - W^c\|^2 = \min_j \gamma_j\|Z(t) - W^j\|^2 \\ -1, & \text{if } i = r \text{ such that } \gamma_r\|Z(t) - W^r\|^2 = \min_{j \ne c} \gamma_j\|Z(t) - W^j\|^2 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where $1 \leq i,\, j \leq K$, and $\gamma_j = \frac{n_j}{\Sigma_{i=1}^{K} n_i}$. n_i is the cumulative frequency of $u_{i(t)} = 1$.

Step 2. Update the centroid W^i and the cumulative frequency n_i by

$$W^i_{\text{new}} = \begin{cases} W^i_{\text{old}} + \alpha_c(Z(t) - W^i), & \text{if } u_{i(t)} = 1 \\ W^i_{\text{old}} - \alpha_r(Z(t) - W^i), & \text{if } u_{i(t)} = -1 \\ W^i_{\text{old}}, & \text{otherwise} \end{cases} \tag{7}$$

$$n_i^{\text{new}} = \begin{cases} n_i^{\text{old}} + 1, & \text{if } u_{i(t)} = 1 \\ n_i^{\text{old}}, & \text{otherwise} \end{cases} \tag{8}$$

where $0 \leq \alpha_r \ll \alpha_c \leq 1$ are the learning rates for the winner and rival cluster nodes respectively (α_r is also called de-learning rate).

The RPCL algorithm can select an appropriate number of cluster nodes automatically by gradually driving extra cluster nodes' centroids far away from the distribution of the data set. Hence, we can remove those extra cluster nodes. In the sequel, as shown in Fig. 2, a cluster layer with the remaining M cluster nodes are produced.

3.2.2. *Local linear predictors*

We assume there exists a function relationship F between $Z_i(\tau)$ and $U_i(\tau)$, where $Z_i(\tau)$ belongs to node C_i and $\{Z_i(\tau),\, U_i(\tau)\}$ is an input-output pair. Hence, by Taylor expansion the local linear predictor attached with cluster node C_i can be represented by:

$$U_i(\tau) \approx F(W^i) + A_i(Z_i(\tau) - W^i)^T \tag{9}$$

where A_i is the first-order derivative of $F(W^i)$. Since $W^i \approx \bar{Z}_i$, we have

$$F(W^i) \approx \bar{U}_i = \Sigma_{j=1}^{n_i} \frac{U_i(\tau)}{n_i} \tag{10}$$

Put Eq. (10) into Eq. (9), the LLP_i in Eq. (9) becomes

$$U_i(\tau) \approx \bar{U}_i + A_i(Z_i(\tau) - W^i)^T \tag{11}$$

where the parameter A_i can be determined by the Least Square method.

3.2.3. *Combination of local linear predictions by gating network*

As shown in Fig. 2, a gating network is applied to combine the local linear predictions $y_{1(t)}, y_{2(t)}, \ldots, y_{k(t)}$ to form $y(t)$ through

$$y_{(t)} = \Sigma_{j=1}^{k} g_{j(t)} y_{j(t)} \tag{12}$$

where $g_{j(t)} = P(j|Z(t))$ is the probability that $Z(t)$ belongs to jth selected cluster node. To estimate it easily, we assume the probability density distribution of $Z(t)$ is the Gaussian:

$$P(Z(t)|m_j) = (2\pi)^{-\frac{d+1}{2}}(|\Sigma_j|)^{-\frac{1}{2}} \times \exp\left\{-\frac{(Z(t) - m_j)^T \Sigma_j^{-1}(Z(t) - m_j)}{2}\right\} \tag{13}$$

where m_j and Σ_j are the jth Gaussian parameters. By Bayes Rule, we can get

$$g_{j(t)} = \frac{\alpha_j P(Z(t)|m_j)}{\Sigma_i \alpha_i P(Z(t)|m_i)} \tag{14}$$

where $\Sigma_j \alpha_j = 1$, $\alpha_j \geq 0$ and $1 \leq i,\, j \leq k$. The parameters Σ_j, m_j and α_j, are obtained by the single-loop EM algorithm as given in Ref. 17. Here, we briefly list the major steps as follows:

- Assume the prediction of each selected cluster nodes is described by the conditional density:

$$P(U(t)|Z(t),\, \sigma^2{}_{yj}) = (2\pi\sigma^2{}_{yj})^{-\frac{1}{2}} \exp\left\{-\frac{(U(t) - y_{j(t)})^2}{2\sigma^2{}_{yj}}\right\} \tag{15}$$

where $\sigma^2{}_{yj}$ is the variance of the output of jth selected cluster node.

- Initialize m_j, Σ_j and α_j. For each iteration r, the following steps are done:
  1. For each t, calculate $P(Z(t)|m_j)$ and $P(U(t)|Z(t), \sigma^2{}_{yj})$ by Eq. (13) and Eq. (15) respectively.
  2. Calculate:

$$\begin{aligned} h_j^{(r)}(U(t)|Z(t)) &= \frac{\alpha_j^{(r)} P(Z(t)|m_j^{(r)}) P(U(t)|Z(t), \sigma_{yj}^2)}{\Sigma_i \alpha_i^{(r)} P(Z(t)|m_i^{(r)}) P(U(t)|Z(t), \sigma_{yj}^2)} \\ m_j^{r+1} &= \frac{1}{\Sigma_t h_j^{(r)}(U(t)|Z(t))} \Sigma_t h_j^{(r)}(U(t)|Z(t)) Z(t) \\ \alpha_j^{(r+1)} &= \frac{1}{N} \Sigma_t h_j^{(r)}(U(t)|Z(t)) \end{aligned} \tag{16}$$

and

$$\Sigma_j^{r+1} = \frac{1}{\Sigma_t h_j^{(r)}(U(t)|Z(t))} \Sigma_t h_j^{(r)}(U(t)|Z(t))(Z(t) - m_j^{r+1})(Z(t) - m_j^{r+1})^T$$

After some iterations of step 1 and step 2, these parameters will converge.[9,16] Then the $g_{j(t)}{}'$s are obtained by putting these parameters into Eq. (13) and Eq. (14).

3.3. *Prediction stage of RPCL-CLP*

In this stage, the input patterns $Z(t)'$s come from testing set and the desired outputs are unknown.

For each $Z(t)$, the Cluster Selector will select k cluster nodes such that the distance between their centroids and $Z(t)$ are the first k smallest ones. We re-denote them by $C'_1, C'_2, \ldots, C'_k$ as shown in Fig. 2. Each of these k attached LLP's gives a prediction denoted by $y_{j(t)}$ through

$$y_{j(t)} = \bar{U}_j + A_j(Z(t) - W^j)^T, \quad j = 1, 2, \ldots, k\,. \tag{17}$$

With the gating network, we obtain the estimation of $U(t)$ by Eq. (12).

4. Adaptive RPCL-CLP Model

4.1. *Data pre-and-post processing*

In Adaptive RPCL-CLP, the involved data pre-and-post processing scheme is specified as:

- Pre-processing $x_t = \ln z_{t+1} - \ln z_t$
- Post-processing $\hat{z}_{t+d+1} = z_{t+d} \times e^{y_{(t)}}$

As a result, the training set and testing set given in Sec. 2 are automatically transformed into new training set $\{(X(t), Y(t))\}_{t=1}^{N-d}$ and new testing set $\{X(t)\}_{t=N-d+1}^{N+S-d-1}$, where at time t we set the input pattern $X(t) = [x_t, x_{t+1}, \ldots, x_{t+d-2}, x_{t+d-1}]$ and the desired output $Y(t) = x_{t+d}$ respectively.

4.2. *Architecture and implementation*

As shown in Fig. 4, the architecture of Adaptive RPCL-CLP is similar to RPCL-CLP except that: (1) a data pre-and-post processing scheme is involved; (2) the Cluster Selector chooses one node only by winner-take-all rule to speed up the training process considerably; (3) the market efficiency is also considered by combining nodes output $y_{m(t)}$ with current input information x_{t+d-1} at time t.

The training procedure of Adaptive RPCL-CLP, which is based on transformed training set, is similar to RPCL-CLP with three steps mentioned in Sec. 3.2. In the prediction stage, Adaptive RPCL-CLP switches to the adaptive learning mode where the parameters in cluster layer and gating network are adaptively adjusted upon each prediction made. In the following subsections, we will describe the adaptive learning algorithm only.

4.2.1. *Adaptive modification of local linear predictors*

In the prediction stage, when an input pattern $X(t)$ extracted from the transformed testing set is presented in the Input Layer II of Adaptive RPCL-CLP, the Cluster Selector will find out a cluster node C_m by winner-take-all rule that satisfies:

$$d_{m(t)} = \min_j \{d_j = \|X(t) - W^j\|, \quad j = 1, 2, \ldots, M\} \tag{18}$$

Hence, the attached LLP prediction of $Y(t)$ is

$$y_{m(t)} = \bar{Y}_m + A_m(X(t) - W^m)^T \tag{19}$$

which is post-processed to become $\hat{z}_{t+d+1}$ by

$$\hat{z}_{t+d+1} = z_{t+d} \times e^{y_{m(t)}} \tag{20}$$

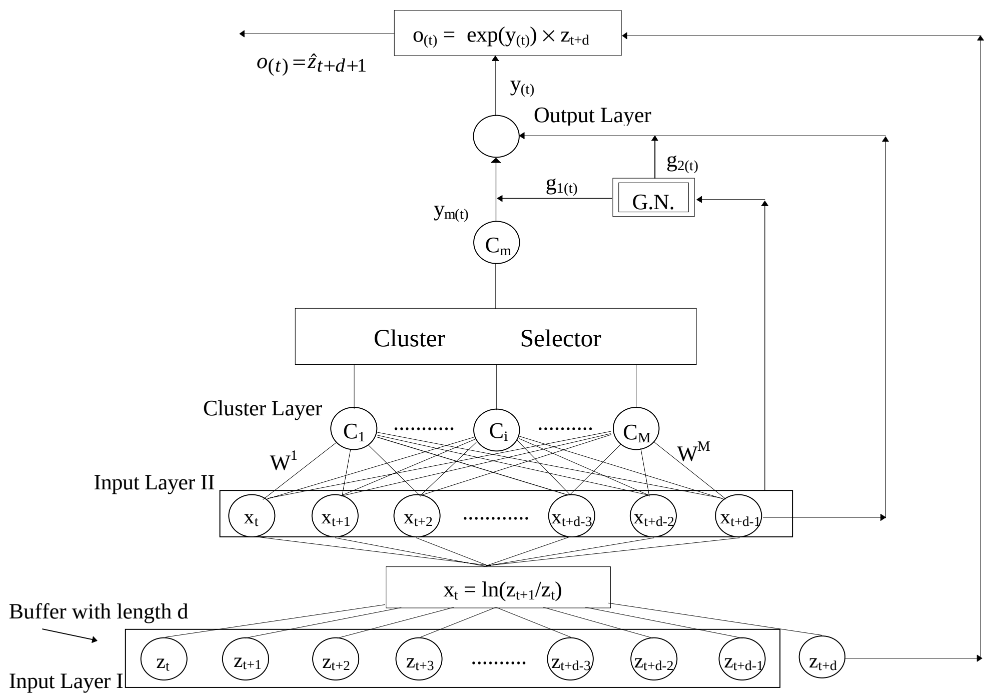


Fig. 4. Architecture of Adaptive RPCL-CLP model, where the cluster nodes C_i's with attached local linear predictors are adaptively learned in the prediction stage as well as the gating network.

As the model performance will degrade if the model parameters are fixed, Adaptive RPCL-CLP adopts an adaptive learning scheme where the parameters of the working node C_m are updated adaptively by the formula:

$$\begin{aligned} n_m^{\text{new}} &= n_m^{\text{old}} + 1 \\ \bar{Y}_m^{\text{new}} &= \bar{Y}_m^{\text{old}} + \eta(Y(t) - \bar{Y}_m^{\text{old}}) \\ W_{\text{new}}^m &= W_{\text{old}}^m + \eta(X(t) - W_{\text{old}}^m) \\ A_m^{\text{new}} &= A_m^{\text{old}} - \frac{1}{2}\eta\frac{\partial \varepsilon_t^2}{\partial P_m} \\ &= A_m^{\text{old}} - \eta[\varepsilon_t(W_{\text{new}}^m - X(t))] \end{aligned} \tag{21}$$

where η is a learning rate, and $\varepsilon_t = (x_{t+d} - y_{m(t)})$.

4.2.2. *Adaptive gating network*

Due to the efficiency of a market as stated in Ref. 1, Random Walk model is useful which uses the current data point x_{t+d-1} as the prediction of the next point. That is,

$$\hat{x}_{t+d} = x_{t+d-1} \tag{22}$$

In Fig. 4, we combine the local linear prediction $y_{m(t)}$ with that of the Random Walk model by a gating network which is implemented by a new adaptive EM algorithm.[15] The detailed algorithm is as follows:

1. At time t of prediction stage, an input pattern $X(t)$ is presented in the Input Layer II. As shown in Fig. 4, we need to combine $y_{m(t)}$ with x_{t+d-1} as $y_{(t)}$. For simplicity, at time t, we re-denote $y_{m(t)}$ as $y_1^{(t)}$, x_{t+d-1} as $y_2^{(t)}$, $g_{1(t)}$ as $g_1^{(t+1)}$ and $g_{2(t)}$ as $g_2^{(t+1)}$ then we have:
$$y_{(t)} = \sum_{i=1}^{2} g_i^{(t+1)} y_i^{(t)} \tag{23}$$
2. To find $g_i^{(t+1)}$ out, we assume that the conditional probability distribution of $Y(t)$ is the Gaussian density:
$$P(Y(t)|m_i^{(t)}, \sigma^2{}_i^{(t)}) = \frac{1}{(2\pi)^{\frac{1}{2}}\sigma_i^{(t)}} \exp\left(\frac{(Y(t) - m_i^{(t)})^2}{2\sigma_i^{2(t)}}\right) \tag{24}$$
where $i = 1$ and 2, $m_i^{(t)}$ and $\sigma^2{}_i^{(t)}$ are the mean and variance of $y_i^{(t)}$ respectively.

Then $g_i^{(t+1)}$ can be obtained from:

$$g_i^{(t+1)} = \frac{P^{(t)}(i)P(Y(t)|m_i^{(t)}, \sigma^{2}{}_i^{(t)})}{\sum_{r=1}^{2} P^{(t)}(r)P(Y(t)|m_r^{(t)}, \sigma^{2}{}_r^{(t)})} \tag{25}$$

Simultaneously, we modify the parameters $m_i^{(t)}$, $\sigma^{2}{}_i^{(t)}$ and $P^{(t)}(i)$ to $m_i^{(t+1)}$, $\sigma^{2}{}_i^{(t+1)}$ and $P^{(t+1)}(i)$ respectively by:

$$h(i|Y(t)) = \frac{g_i^{(t+1)} - g_i^{(t)}}{t}$$

$$P^{(t+1)}(i) = P^{(t)}(i) + h(i|Y(t))$$

$$\alpha_i = \frac{P^{(t)}(i)}{P^{(t+1)}(i)} \quad \text{and} \quad \beta_i = \frac{1}{P^{(t+1)}(i)}$$

$$m_i^{(t+1)} = \alpha_i m_i^{(t)} + \beta_i h(i|Y(t))Y(t)$$

$$\sigma^{2}{}_i^{(t+1)} = \alpha_i \sigma^{2}{}_i^{(t)} + \beta_i h(i|Y(t))[Y(t) - m_i^{(t)}]^2 \tag{26}$$

As pointed out in Ref. 15, we can get a better estimate for $g_i^{(t)}$ by using Eq. (23) again as long as a set of the second latest values $m_i^{(t-1)}$, $\sigma^{2}{}_i^{(t-1)}$, $P^{(t-1)}(i)$ are kept in memory, which is a key difference between the adaptive EM algorithm given in Ref. 15 and the incremental EM algorithm given in Ref. 12.

5. Trading Model

A trading system can regulate the trading activities when the present information is provided. In real life, its underlying mechanism may be far too complicated. In this paper, we propose a simple system comprising a prediction model and a trading model as shown in Fig. 5.

The trading model applies one of three different risk-level trading strategies, each of which has different rules to determine when to buy a long (or short) contract and balance the contract. Transaction costs and interest gains are ignored for simplicity in the implementation of our trading strategies. The three trading strategies are specified as follows:

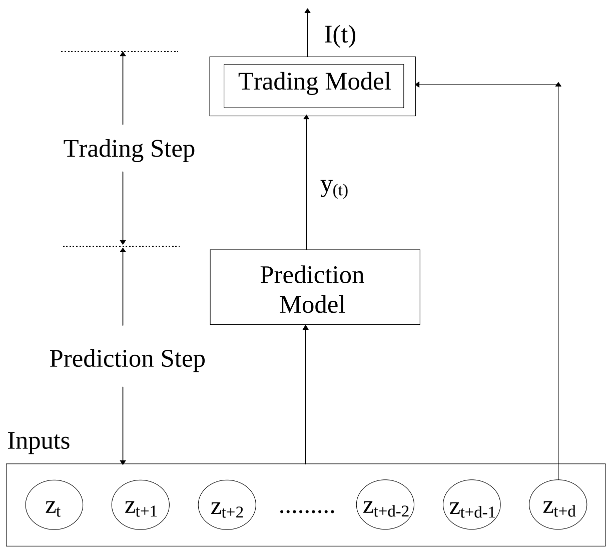


Fig. 5. Trading system with two steps: prediction step followed by trading step. In prediction step, the prediction model outputs $y_{(t)}$ — the estimation of desired output $U(t)$ when the input $Z(t) = [z_t, z_{t+1}, \ldots, z_{t+d-1}, z_{t+d}]$ is available. In trading step, the trading model will output a trading signal $I(t)$ based on $y_{(t)}$ and current information z_{t+d} for the next trading activity.

Strategy 1

1. At the beginning, given the current price z_t and the prediction $\hat{z}_{t+1}$ of the next price, an trading indicator $I(t)$ of the trading system emits one of the three signals: -1, 0 and 1, determined by

$$I(t) = \begin{cases} -1, & \text{if } \hat{z}_{t+1} - z_t > 0 \\ 0, & \text{if } \hat{z}_{t+1} - z_t = 0 \\ 1, & \text{otherwise} \end{cases} \tag{27}$$

where $I(t) = 1$, 0 and -1 stand for the "buy long," "do nothing" and "buy short" signals respectively.

2. After we have bought a long (short) contract, we update the Gains by:

$$\text{Gains}^{(\text{new})} = \text{Gains}^{(\text{old})} + \begin{cases} -\text{diff}, & \text{if long contract has been bought} \\ \text{diff}, & \text{if short contract has been bought} \end{cases} \tag{28}$$

where $\text{diff} = z_{t+1} - z_t$. Simultaneously, the profit R is determined by

$$R^{(\text{new})} = R^{(\text{old})} + \text{Gains}^{(\text{new})} \tag{29}$$

Since we can predict the difference $(\hat{z}_{t+1} - z_t)$ when z_t is available, if the predicted trend is not good to hold the contract or the returns have exceeded a pre-defined threshold value γ (the desired profits in each transaction), we have to determine whether we need to balance the contract according to the decision Indicator $I_d(t)$ below:

$$I_d(t) = \begin{cases} 1, & \text{if } (R > \gamma) \text{ or } (|\hat{z}_{t+1} - z_t| > r \times R) \\ 0, & \text{otherwise} \end{cases} \tag{30}$$

where $I_d(t) = 1$ and 0 stand for "sell" and "do nothing" signals respectively. r is a risk factor with $0 \leq r \leq 1$ that is set at 0.5 in the experiment.

3. If the transaction is not balanced, we go to step (2). Otherwise, before returning to step (1), we update the gained points (1 Point = 0.0001 DEM) according to the formula

$$\text{Points}^{(\text{new})} = \text{Points}^{(\text{old})} + 10000 \times \text{Gains}^{(\text{new})} \tag{31}$$

and reset the Gains to zero.

Strategy 2

Strategy 2 is a variant of Strategy 1 with the same procedure except that the formula in Eq. (29) becomes:

$$R^{(\text{new})} = \text{Gains}^{(\text{new})} \tag{32}$$

Strategy 3

Strategy 3 is also the same as Strategy 1 except that

1. the trading indicator $I(t)$ in Eq. (27) becomes:

$$I(t) = \begin{cases} -1, & \text{if } (\hat{z}_{t+1} - z_t \geq c) \\ 0, & \text{if } (|\hat{z}_{t+1} - z_t| \geq c) \\ 1, & \text{otherwise} \end{cases} \tag{33}$$

2. the profit R is determined by Eq. (32) instead of Eq. (29).

where $c = (l\max - l\min)$days_no, $l\max(l\min)$ is the maximum (minimum) of z_t in the last days_no days before testing.

Before closing this section, we want to point out two things:

1. Among three trading strategies mentioned above, Strategy 1 has the highest investment risk, and Strategy 3 has the lowest.
2. If the prediction follows the Random Walk, the above trading model can also work as long as z_t is replaced by z_{t-1}.

6. Computer Experiments

6.1. *Data sets and experimental purpose*

Two real-world financial time series are used for financial prediction. One is a China stock from December 21, 1990 to September 12, 1995. The training data set (training set I) is the first 1100 data points whereas the testing data set (testing set I) is the remaining 90 data points.

The other series is an exchange-rate series of US Dollar (USD) versus Deutschmark (DEM), which consists of 1778 daily closing prices from December 1, 1987 to November 30, 1993. As we only have the data of DEM versus Hong Kong dollars (HKD) and USD versus HKD, we transform these data to USD versus DEM by the formula: 1 USD = (m/n) DEM, where m is the rate of USD versus HKD and n is the rate of DEM versus HKD. The training set (training set II) is the first 1679 data, and the testing set (testing set II) is the following 99 data points.

There are two purposes in our experiments:

1. We compare the performance of Adaptive RPCL-CLP and RPCL-CLP with some existing models namely Recurrent Networks (Elman Net and Jordan Net), MA(q) model (q is often set at 5 or 10 in practice) and Random Walk model. We calculate the prediction error and the profit gains in the trading simulation of foreign exchange market, where we assume at a time a trader can only hold at most one long (or short) contract of USD versus DEM valued 50,000 DEM with deposit of US$6500 (the lowest marginal deposit required by the Hong Kong Financial Bureau).
2. We justify the efficacy of our data pre-and-post processing scheme by comparing the recurrent nets with and without this scheme.

6.2. *Experimental results*

Figures 6–8 show the results of Adaptive RPCL-CLP, RPCL-CLP, Elman Net and Jordan Net on

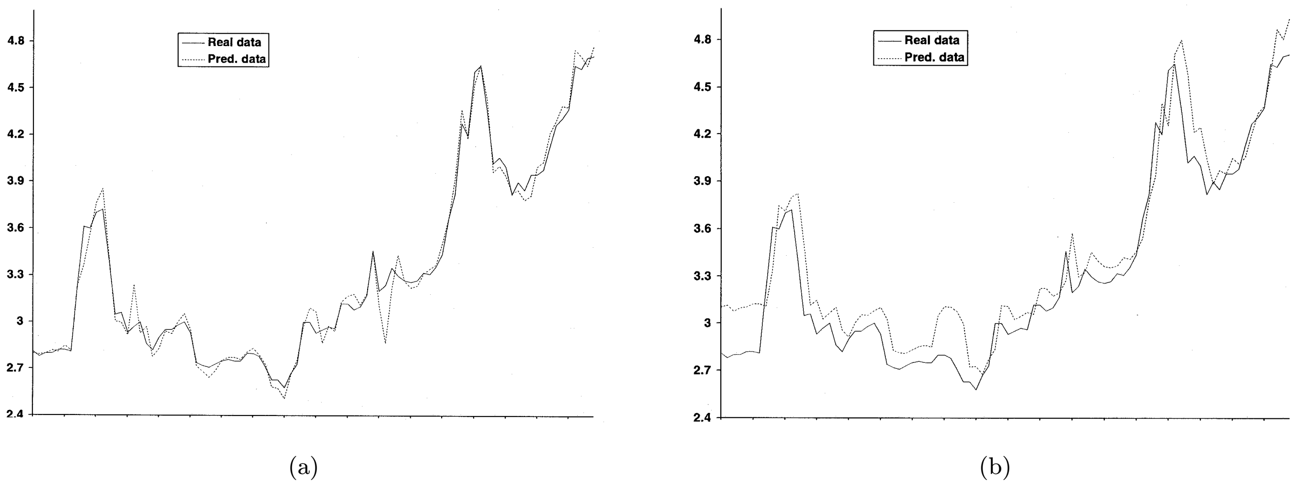


Fig. 6. The prediction results on testing set I, (a) by Adaptive RPCL-CLP model; (b) by RPCL-CLP model.

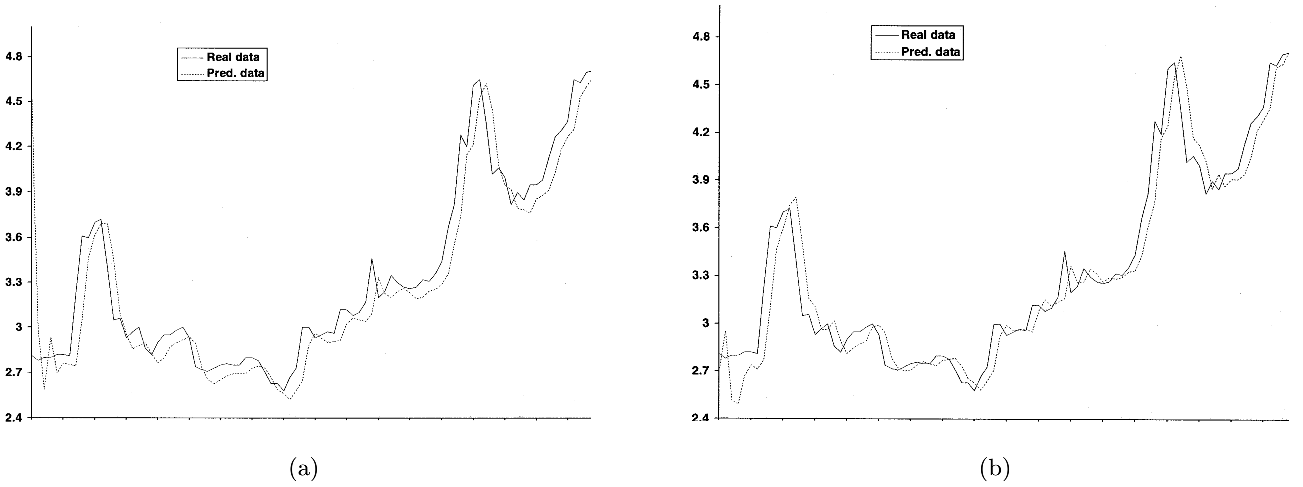


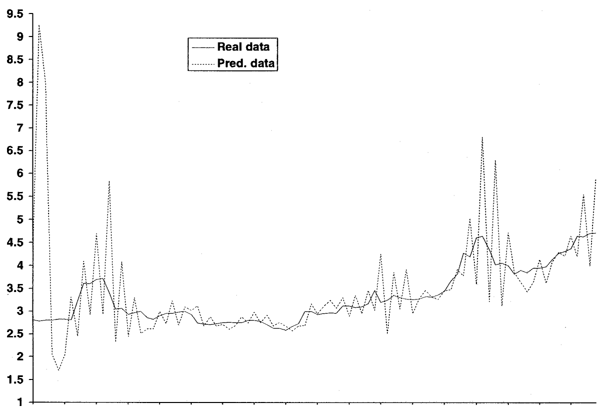
Fig. 7. Elman Net on testing set *I*, (a) without data pre-and-post processing scheme; (b) with data pre-and-post processing scheme.

testing set I whereas Figs. 9–11 show their results on testing set II. The parameters of Elman and Jordan Nets trained by the back-propagation with momentum algorithm[13] are adjusted by trial and error. Tables 1 and 2 list the training costs and prediction errors in terms of *root mean square error* (r.m.s.e.) as well as the number of hidden nodes. Furthermore, Table 3 also lists the results of MA(q) models as well as Random Walk model for testing set I and II respectively.

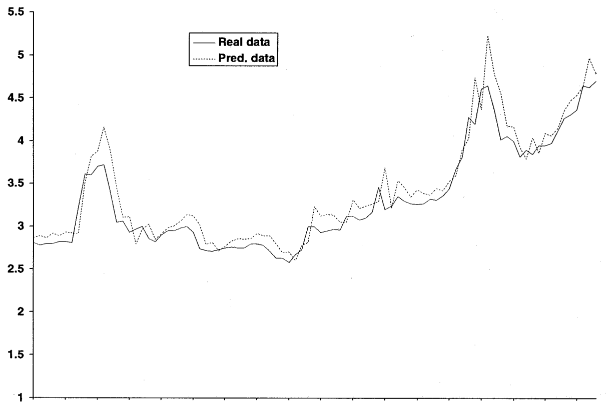
Figures 12–17 show the simulation results in the foreign exchange market, where the predictions of the USD versus DEM series are given by the prediction models mentioned above. The gains are calculated by the formula: 1 Point $\cong$ US\$6.5. Tables 4–6 show the transaction times in the trading process with the statistical time interval of 15 days.

In the experiments, we can observe the following points:

1. The proposed Adaptive RPCL-CLP outperforms the existing models mentioned above in terms of prediction error. There are three reasons:
   (a) The cluster nodes in Adaptive RPCL-CLP separate the input space into a set of local regions. At one time a local linear predictor LLP in one of the regions is triggered by the Cluster Selector. That is, those

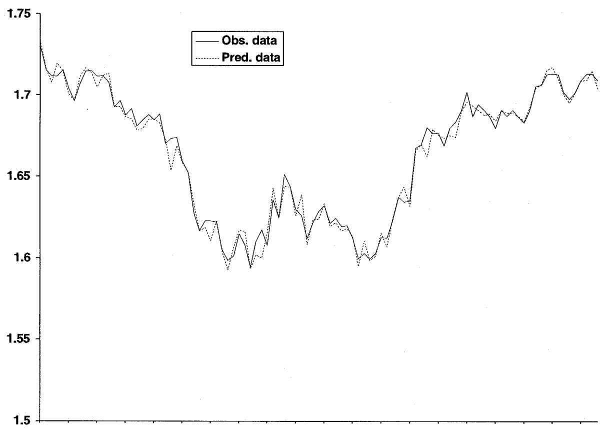


(a)

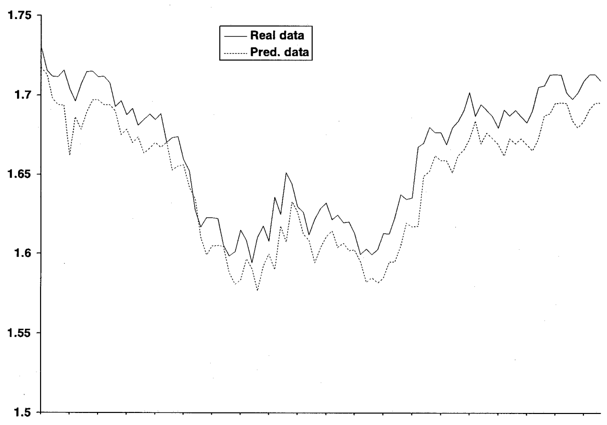


(b)

Fig. 8. Jordan Net on testing set I, (a) without data pre-and-post processing scheme; (b) with data pre-and-post processing scheme.



(a)



(b)

Fig. 9. The prediction results on testing set II, (a) by Adaptive RPCL-CLP model; (b) by RPCL-CLP model.

LLP's can be regarded as local experts which are gated by the Cluster Selector. However, these existing models do not have mixture-of-experts mechanism.

(b) Unlike Adaptive RPCL-CLP, the parameters of Elman Net and Jordan Net cannot be globally determined due to their high non-linearity.

(c) The Adaptive RPCL-CLP applies an adaptive learning scheme to adjust the model parameters.

2. Data pre-and-post processing can considerably improve the models performance as shown in Figs. 7 and 8 and Figs. 10 and 11.
3. Using the same prediction model, our profit gains are determined by the trading strategies. As shown in Fig. 18, Strategy 1 whose trading risk is the highest can obtain the largest profits. Even in the same trading strategy, the risk factor r also affects the profit gains as shown in Fig. 19.
4. Random Walk model can forecast the two financial series well with fairly small r.m.s.e.

7. Conclusion

As shown by our experiments on the financial prediction and trading simulation, Adaptive RPCL-CLP

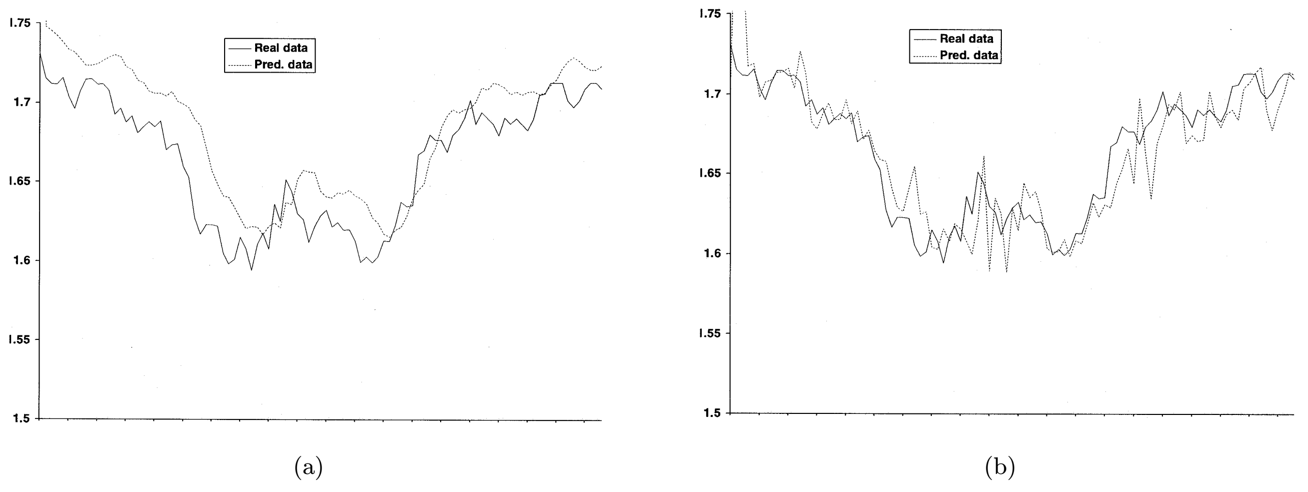


Fig. 10. Elman Net on testing set II, (a) without data pre-and-post processing scheme; (b) with data pre-and-post processing scheme.

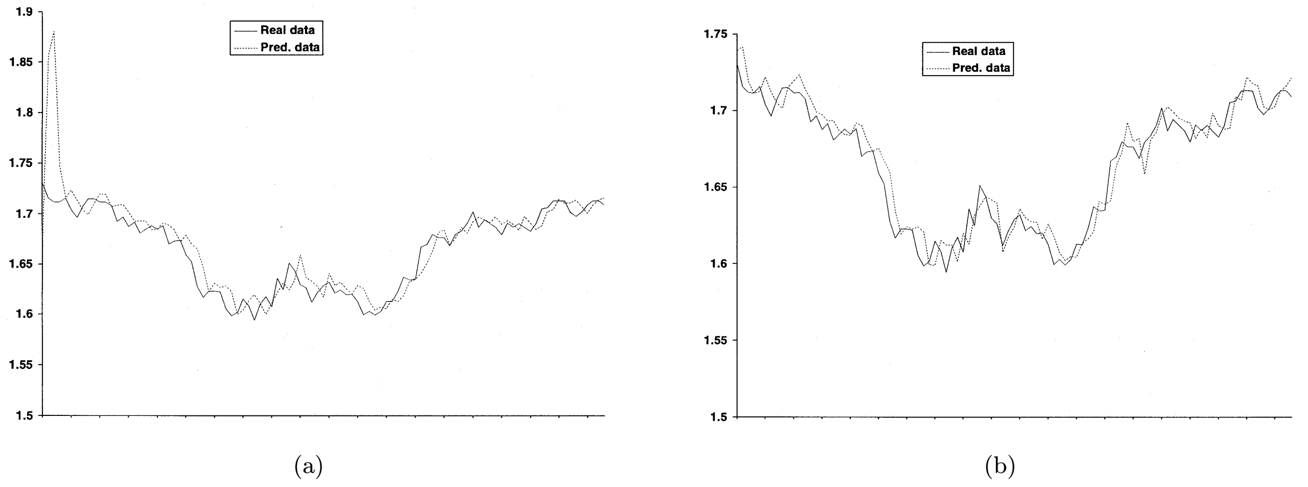


Fig. 11. Jordan Net on testing set II, (a) without data pre-and-post processing scheme; (b) with data pre-and-post processing scheme.

Table 1. The results of Adaptive RPCL-CLP with RPCL-CLP, Elman Net and Jordan Net on data set I.

Models	Flops (Training Set I)	No. of Hidden Nodes	(Testing Set I)
Adaptive RPCL-CLP	4.100×10^6	73	0.0790
RPCL-CLP	5.260×10^6	21	0.1787
Elman Net[1]	1.736×10^{10}	40	0.2557
Elman Net[2]	8.706×10^9	40	0.1726
Jordan Net[1]	5.276×10^9	40	1.0899
Jordan Net[2]	8.792×10^9	40	0.2004

Note: Data set I consists of training set I and testing set I. Elman[1] denotes the Elman net without data pre-and-post processing scheme, Elman[2] denotes the Elman net with data pre-and-post processing scheme, Jordan[1] and Jordan[2] are denoted in the similar way. Training costs are measured in terms of flops in MATLAB.

Table 2. The results of Adaptive RPCL-CLP with RPCL-CLP, Elman Net and Jordan Net on data set II.

Models	Flops (Training Set II)	No. of Hidden Nodes	r.m.s.e. (Testing Set II)
Adaptive RPCL-CLP	8.789×10^6	120	0.0053
RPCL-CLP	1.235×10^7	8	0.0205
Elman Net[1]	1.356×10^{10}	40	0.0378
Elman Net[2]	3.984×10^9	20	0.0230
Jordan Net[1]	1.751×10^{10}	45	0.0265
Jordan Net[2]	1.396×10^{10}	40	0.0111

Note: Data set II consists of training set II and testing set II.

Table 3. The prediction errors of MA(q) and Random Walk models on testing set I and II.

	r.m.s.e. (Testing Set I)	r.m.s.e. (Testing Set II)
Random Walk	0.1423	0.0098
MA(5)	0.2566	0.0144
MA(10)	0.3248	0.0203

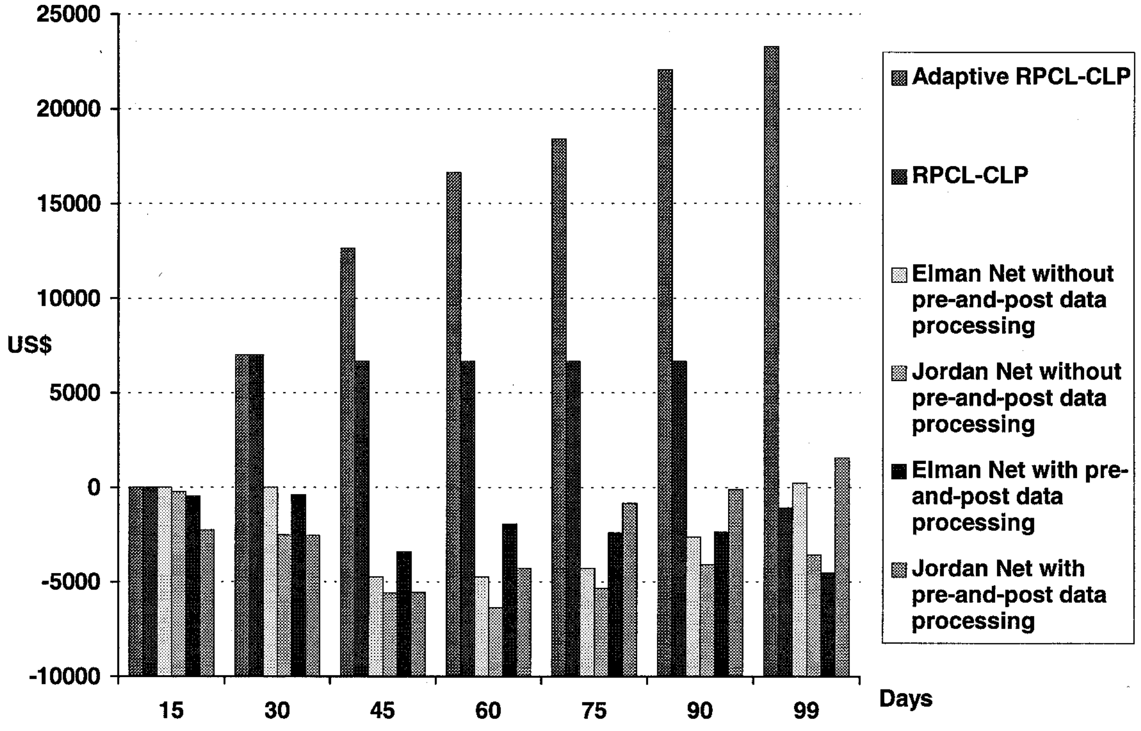


Fig. 12. Profits from different prediction models with trading strategy 1.

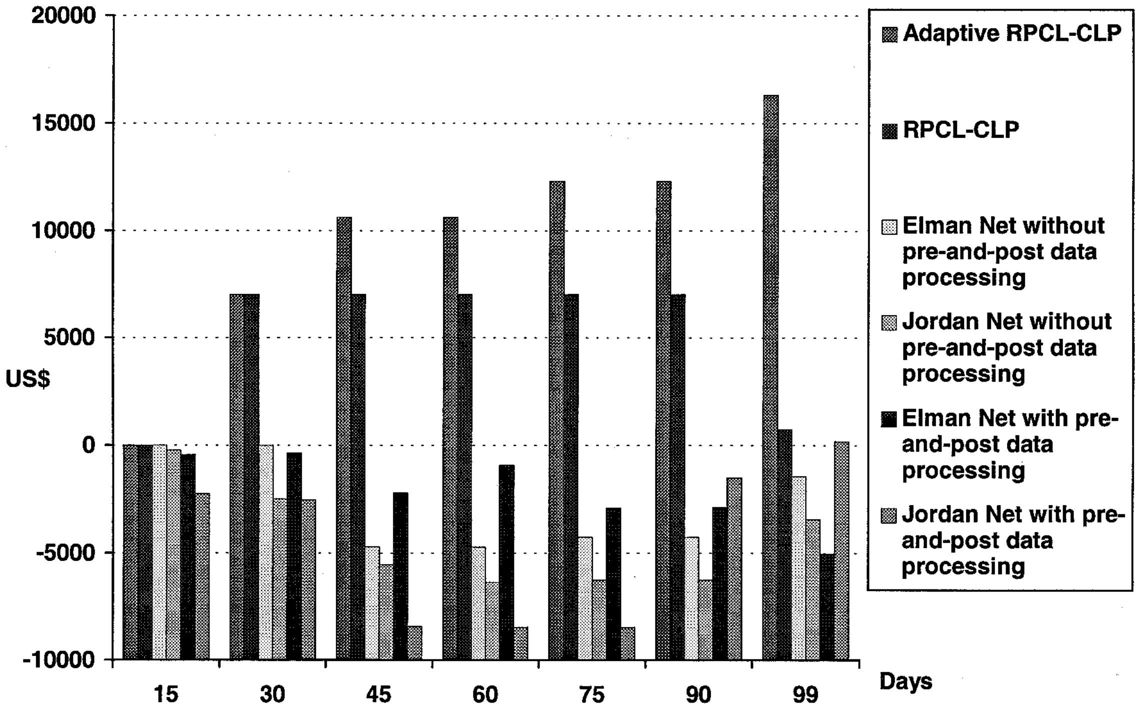


Fig. 13. Profits from different prediction models with trading strategy 2.

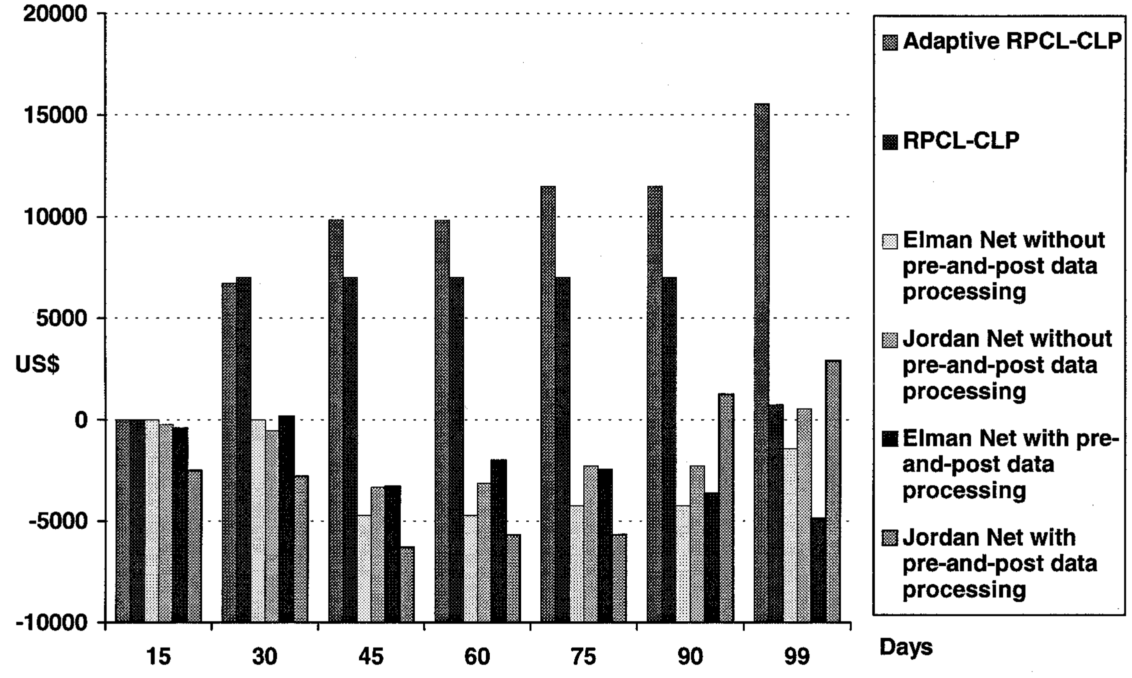


Fig. 14. Profits from different prediction models with trading strategy 3.

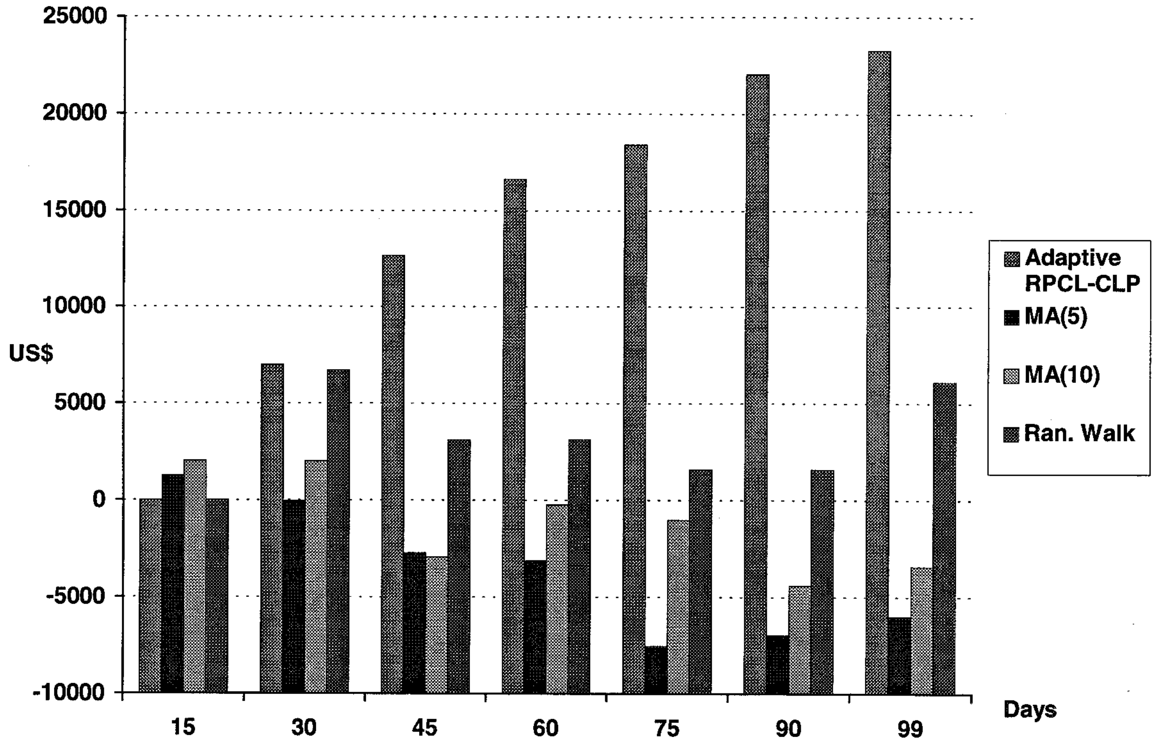


Fig. 15. Profits from different prediction models with trading strategy 1.

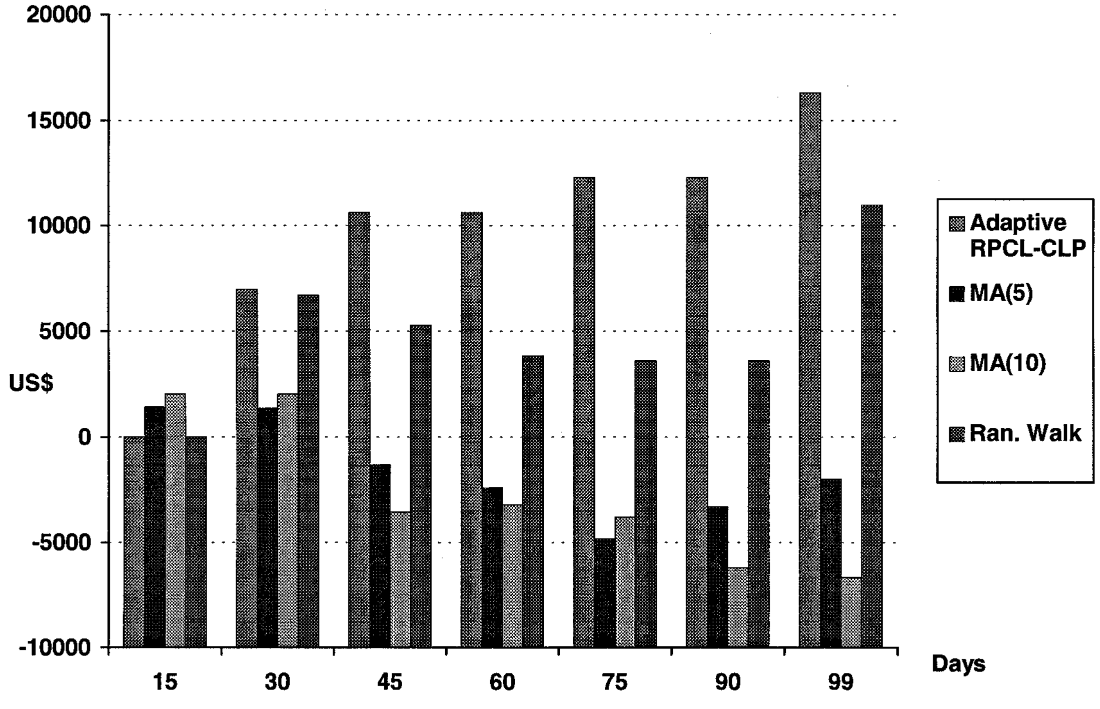


Fig. 16. Profits from different prediction models with trading strategy 2.

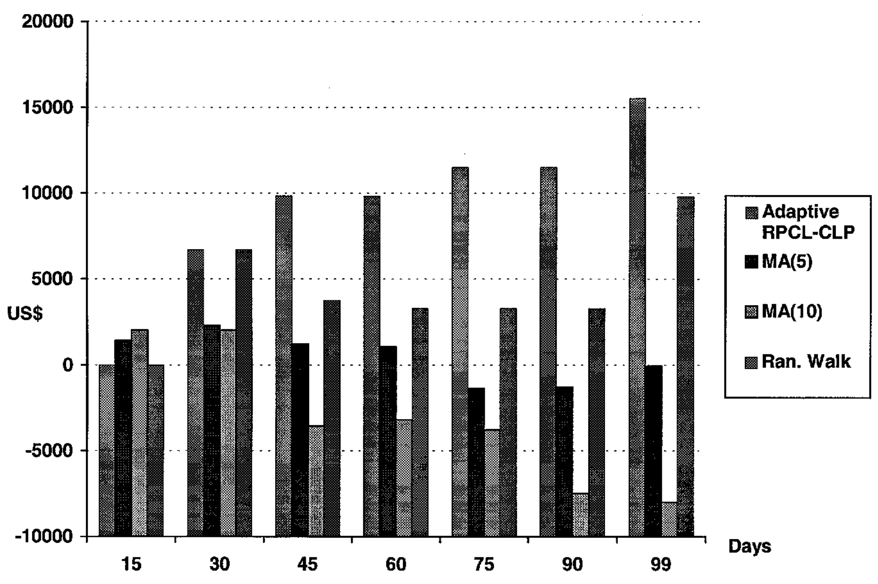


Fig. 17. Profits from different prediction models with trading strategy 3.

Table 4. The transaction times under trading strategy 1 with $r = 0.5$ and $\gamma = 0.1$.

	Days						
Models	15	30	45	60	75	90	99
Adaptive RPCL-CLP	1	3	18	33	48	63	72
RPCL-CLP	1	3	11	11	11	11	12
Elman Net[1]	1	1	4	5	13	15	16
Elman Net[2]	8	17	22	34	41	51	58
Jordan Net[1]	9	15	21	27	38	49	58
Jordan Net[2]	7	11	20	31	37	43	44
MA(5)	3	7	11	17	21	31	36
MA(10)	3	3	7	13	15	23	28
Random Walk	1	3	10	11	13	13	14

Table 5. The transaction times under trading strategy 2 with $r = 0.5$ and $\gamma = 0.1$.

	Days						
Models	15	30	45	60	75	90	99
Adaptive RPCL-CLP	1	3	17	17	21	21	22
RPCL-CLP	1	3	3	3	3	3	4
Elman Net[1]	1	1	4	5	13	13	14
Elman Net[2]	8	17	22	29	37	47	54
Jordan Net[1]	9	15	21	27	37	37	38
Jordan Net[2]	7	11	17	21	21	29	30
MA(5)	7	11	15	19	21	31	36
MA(10)	3	3	5	7	9	17	18
Random Walk	1	3	13	25	29	29	30

Table 6. The transaction times under trading strategy 3 with $r = 0.5$ and $\gamma = 0.1$.

	Days						
Models	15	30	45	60	75	90	99
Adaptive RPCL-CLP	1	2	13	13	17	17	18
RPCL-CLP	1	3	3	3	3	3	4
Elman Net^1	1	1	4	5	13	13	14
Elman Net^2	8	15	20	26	33	41	44
Jordan Net^1	9	13	19	25	33	33	34
Jordan Net^2	7	11	17	21	21	26	28
MA(5)	7	11	15	19	21	25	30
MA(10)	3	3	5	7	9	11	12
Random Walk	1	3	15	23	25	25	26

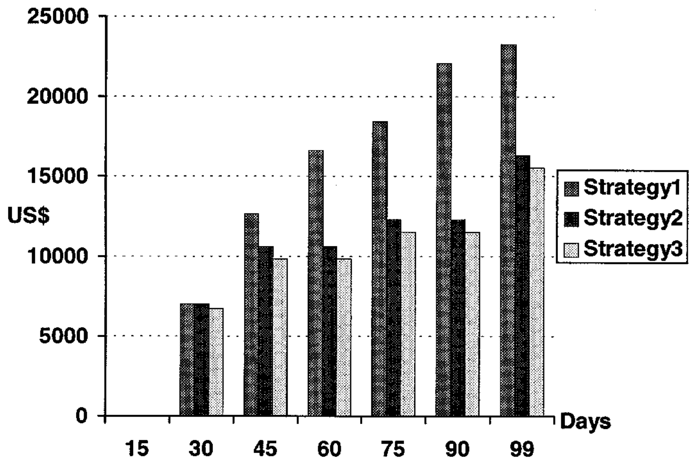


Fig. 18. Profits from Adaptive RPCL-CLP with different trading strategies.

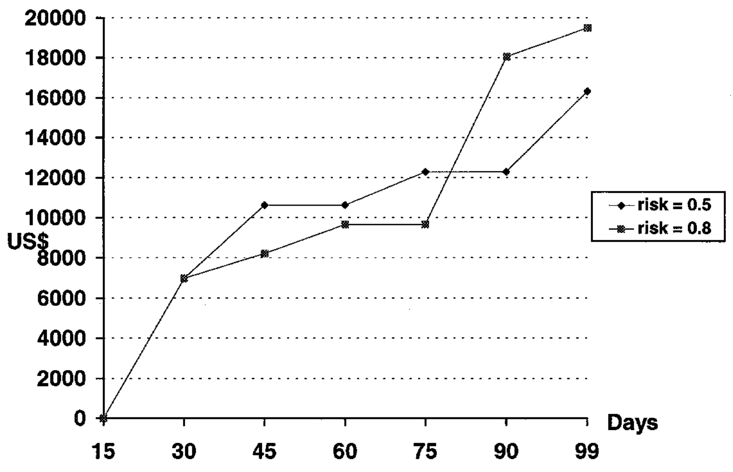


Fig. 19. Profits from Adaptive RPCL-CLP with different risk factors.

is superior to RPCL-CLP, Elman net, Jordan net, MA(q) and Random Walk models in terms of prediction errors as well as profit gains. Furthermore, Adaptive RPCL-CLP is trained much faster than Elman and Jordan nets.

In this paper, we also found that a data pre-and-post processing scheme can considerably improve the model's performance. However, to design a more powerful scheme, further studies are still required.

Acknowledgments

We are very grateful to Ms. Zhi-hong Lai for fruitful discussion on the trading system in Sec. 5 and providing us with the experimental data.

This work was supported by the HK RBC Earmarked Grants CUHK 250/94E, CUHK 484/95E and by Ho-Sin-Hang Education Endowment Fund for Project HSH 95/02.

References

1. P. Cootner 1964, *The Random Character of Stock Market Prices* (MIT Press, Cambridge, Mass).
2. J. L. Elman 1990, "Finding structure in time," *Cognitive Science* **14**, 179–211.
3. D. Farmer and J. J. Sidorowich 1987, "Predicting chaotic time series," *Phys. Rev. Lett.* **59**(8), 845–848.
4. N. A. Gershenfeld and A. S. Weigend 1993, "The future of time series: Learning and understanding," in *Time Series Prediction: Forecasting the Future and Understanding the Past*, eds. A. S. Weigend and N. A. Gershenfeld (SFI Studies in the Sciences of Complexity, Addison-Wesley), pp. 1–70.
5. M. I. Jordan 1986, "Attractor dynamics and parallelism in a connectionist sequential machine," in *Proc. Eighth Ann. Conf. Cognitive Science Society*, Amherst 1986, (Erlbaum, Hillsdale), pp. 531–546.
6. M. I. Jordan 1988, "Supervised learning and systems with excess degrees of freedom," *COINS Technical Report 88–27*, Massachusetts Institute of Technology.
7. M. I. Jordan 1989, "Serial order: A parallel, distributed processing approach," in *Advances in Connectionist Theory*: *Speech*, eds. J. L. Elman and D. E. Rumelhart (Erlbaum, Hillsdale).
8. A. M. I. Jordan and R. A. Jacobs 1990, "Learning to control an unstable system with forward modelling," *Advances in Neural Information Processing 2*, ed. D. Touretzky (Morgan Kaufmann, San Mateo), pp. 324–331.
9. M. I. Jordan and L. Xu 1995, "Convergence results for EM approach to mixtures of experts architectures," *Neural Networks* **8**(9), 1409–1431.
10. A. Lapedes and R. Farber 1987, "Nonlinear signal prediction using neural networks: Prediction and system modelling," preprint, Los Alamos National Laboratory LA-UR-87-2662.
11. M. C. Mozer 1993, "Neural net architectures for temporal sequence processing," in *Time Series Prediction: Forecasting the Future and Understanding the Past*, eds. A. S. Weigend and N. A. Gershenfeld (SFI Studies in the Sciences of Complexity, Addison-Wesley), pp. 243–265.
12. R. N. Neal and G. E. Hinton 1993, "A new view of the EM algorithm that justifies incremental and other variants," CS Department, University of Toronto, pre-print.
13. D. Plaut, S. Nowlan and G. Hinton 1986, "Experiment on learning by back propagation," *Technical Report CMU-CS-86-126*, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA.
14. E. A. Wan 1993, "Time series prediction by using a connectionist network with delay lines," in *Time Series Prediction: Forecasting the Future and Understanding the Past*, eds. A. S. Weigend and N. A. Gershenfeld (SFI Studies in the Sciences of Complexity, Addison-Wesley), pp. 195–207.
15. L. Xu 1995, "YING-YANG machine: A Bayesian-Kullback scheme for unified learnings and new results on vector quantization," *Invited Paper, Proc. Int. Conf. Neural Information Processing* (*ICONIP'95*), Beijing, China, Vol. 2, pp. 977–988.
16. L. Xu, M. I. Jordan 1996, "On convergence properties of the EM algorithm for Gaussian mixtures," *Neural Comput.* **8**(1), 129–151.
17. L. Xu, M. I. Jordan and G. E. Hinton 1994, "A modified gating network for the mixtures of experts architecture," in *Proc. World Cong. Neural Networks* **2**, 405–410.
18. L. Xu, A. Krzyzak and E. Oja 1993, "Rival penalized competitive learning for clustering analysis, RBF net, and curve detection," *IEEE Trans. Neural Networks* **4**, 636–648.