Neural Processing Letters **18:** 197–211, 2003.

Improved CBP Neural Network Model with Applications in Time Series Prediction

DAI QUN, CHEN SONGCAN and ZHANG BENZHU
Department of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

Abstract. Circular back-propagation neural network (CBP) put forward by Sandro Ridella and Stefano Rovetta, a generalized model of multi-layer perceptron (MLP), possesses strong capabilities of generalization and adaptation to unknown inputs. And they can flexibly construct vector quantization (VQ) and radial basis function (RBF) networks under the CBP framework. With the original structure of CBP remaining unchanged, in this Letter a more generalized network model ICBP (Improved Circular Back-Propagation Neural Network) was designed by adding an extensive node with quadratic form to the original CBP inputs and endowing fixed values to the weights between this node and all the hidden nodes. An interesting property of ICBP is that although it has less adaptable weights, it is better in generalization and adaptability than CBP. Moreover, in order to partially solve the problem of local minima, we adopt the method of adding controlled noise to desired outputs. Finally, it has been proved by experiments that ICBP is better than CBP in the capabilities of forecasting and function approximation.

Key words. circular back-propagation neural network, improved circular back-propagation neural network, neural network, radial basis function network, time series prediction

1. Introduction

The learning algorithm of error back-propagation (EBP) is one of the most frequently applied algorithms in the study of artificial neural networks (ANN). On its basis, Sandro Ridella and Stefano Rovetta presented circular back-propagation (CBP) network. CBP adds to its input layer an extra node having an input of the sum of the squared input components [1, 2]. With the same architecture of MLP, CBP possesses the following merits: (1) CBP for pattern recognition can switch automatically between prototype-based and surface-based classifiers; (2) CBP takes RBF network as its special example. Moreover, it is better than RBF in function approximation; (3) CBP bears clearer explanations to knowledge; (4) It exhibits not only the globality of MLP but also the locality of RBF; (5) RBF can be constructed in the CBP frame, but CBP can not be realized in the structure of RBF; (6) The leaning algorithm of CBP is the same as BP, so all algorithms modified to BP can be applied to CBP; (7) CBP possesses better generalization capability than MLP. Despite so, CBP has the following shortages: (1) The extra input item can only represent isotropy, but not anisotropy. Therefore, the representation capability of CBP is limited; (2) Also due to isotropy, CBP can not realize Bayesian minimum error classifier with unequal

covariance; (3) In order to arbitrarily approximate a function, there must be adequate hidden nodes in CBP, which will produce redundant adjustable parameters. This will possibly result in over-fitting and generalization capability decline [1].

Retaining the original structure of CBP, we obtain a more general network model-ICBP through a special but simple construction to the extra node in the CBP input layer and special value (+1 or −1, etc) assigned to the weights between the node and the hidden layer. Compared with CBP, ICBP has the following virtues: (1) It has less adaptable weights but better generalization and adaptation; (2) ICBP has the characteristic of anisotropy; (3) 2^{N_h} kinds of ICBP networks with different characteristics can be transformed from 2^{N_h} combination of different assignments to the weights between the extra node and the hidden layer; (4) The BP learning algorithm can be adopted. Therefore, all the improvements made to BP can boost ICBP and CBP performances; (5) It is more general than CBP, and takes CBP as a special case.

The emphases of this work are on the time series prediction quality comparison between CBP and ICBP. Thus their learning algorithms still adopt the BP algorithm. In order to partially cope with its local minima problem, we train ICBP and CBP by adding controlled noise to the desired outputs.

The forecasting experiments on the chaotic time series, multiple-input multiple-output (MIMO) systems and the data sets of daily life water consumed quantity have proved that ICBP has better capabilities of prediction and approximation than CBP. Although the advancements of ICBP are direct and natural, its excellent properties of constructive equivalence to RBF (refer to Appendix 1) make its application broader and more practical.

2. Improved CBP Neural Network Model

2.1. STRUCTURE AND PROPERTIES OF ICBP

Figure 1 shows a three-layer ICBP network with entirely the same structure of CBP. It has N_o output nodes, N_h hidden nodes, d input nodes with respect to d dimensional input pattern. And it has an extra input node with the input being $x_{d+1} = \sum_{i=1}^{d} a_i^2 x_i^2$, while in CBP $x_{d+1} = \sum_{i=1}^{d} x_i^2$ instead. Therefore when all a_i are taken equal, ICBP reduces to CBP. And at the same time, ICBP weights connecting the extra node to all the nodes in the hidden layer differ from CBP ones: $v_{j(d+1)}(j = 1, \ldots, N_h)$ for ICBP take a common constant directly while the counterparts are adaptable parameters. Consequently, the discrepancy of the number of adaptable parameters for these two models is $|N_h - d|$. In general, the number of hidden nodes is larger than input nodes due to the proven result that the forward multi-layer networks with sufficient hidden nodes number can approximate any continuous function to arbitrary precision. Therefore, the adjustable parameters of ICBP are often less than CBP.

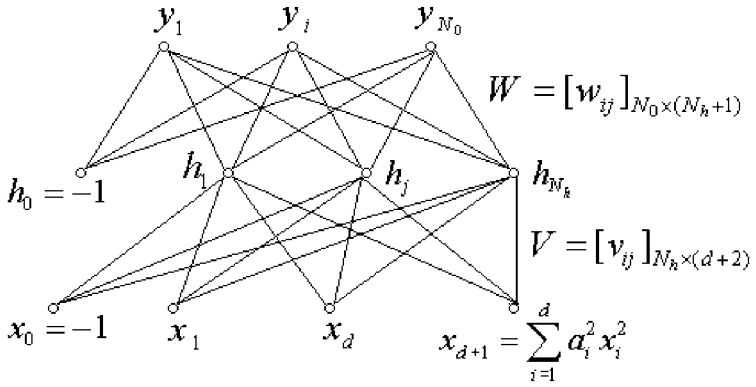

Figure 1. ICBP three-layer network model.

For convenience, below given are some notations used in the Letter:

$(x_0, x_1, \ldots, x_d)$: the network inputs.
x_{d+1}: the extra input as defined above.
V: the weight matrix between the input and hidden layer.
W: the weight matrix between the hidden and output layer.
$y_i = \sum_{j=0}^{N_h} w_{ij} h_j, i = 1, \ldots, N_o$: the i-th output of network.
$h_j = \sigma(r_j) = 1/(1 + e^{-r_j}), j = 1, \ldots, N_h$: the activation output of the jth hidden node, where r_j denotes the following sum of weighted inputs:

$$r_j(\vec{x}, \vec{v}) = \sum_{i=0}^{d} v_{ji} x_i + v_{j(d+1)} x_{d+1} = -v_{j0} + \sum_{i=1}^{d} (v_{ji} x_i + v_{j(d+1)} a_i^2 x_i^2) \tag{1}$$

From Equation (1), if we assign all $v_{j(d+1)}$ as $+1$, then ICBP extends the spherical activation fields of the neurons in the CBP hidden layer to quadratic hyper-ellipsoidal activation fields; if we set $v_{j(d+1)}$ as $+1$ or -1 alternatively, then the activation field of the neurons in the ICBP hidden layer is hyperboloid. Actually there exist 2^{N_h} different assignments of $+1$ or -1 to $v_{j(d+1)} (j = 1, \ldots, N_h)$ and thus produce 2^{N_h} different ICBP network models.

After adding a sum of squared input components to the regular neuron input and $v_{jd+1} a_i^2 \neq 0, (i = 1, \ldots, d)$, we have:

$$r_j(\vec{x}, \vec{v}) = v_{jd+1}[(\vec{x} - \vec{c}_j)\Lambda(\vec{x} - \vec{c}_j)^T - \theta_j] \quad \Lambda = \begin{bmatrix} a_1^2 & & & 0 \\ & a_2^2 & & \\ & & \ddots & \\ 0 & & & a_d^2 \end{bmatrix} \tag{2}$$

where $\vec{c}_j = -\frac{1}{2v_{jd+1}}(v_{j1}/a_1^2, v_{j2}/a_2^2, \ldots, v_{jd}/a_d^2)$ is the center of a hyper-ellipsoid, $\theta_j = (\vec{c}_j \Lambda \vec{c}_j{}^T + v_{j0} / v_{jd+1})$ a threshold, and Λ is a diagonal and positive definite matrix. An operation explanation of the formula is: For each hidden unit, it firstly calculates

the ellipsoidal distance between sample $\vec{x}$ and $\vec{c}_j$, and subtracts θ_j, then multiplies coefficient $v_{j(d+1)}$, and finally outputs an activated values h_j through the function σ. When the sample is in the hyper-ellipsoid field determined by $\vec{c}_j$, Λ and $\sqrt{\theta_j}$, set $v_{jd+1} < 0$; when it lies outside this hyper-ellipsoid field, set $v_{jd+1} > 0$. Thereby it is proven that the output is always positive in the hidden layer, which means the model can realize prototype-based classification. However, when setting all the coefficients $a_i^2 = 0, (i = 1, \ldots, d)$ in Equation (1), ICBP degenerates to the usual BP model, which means that a surface-based classification can be realized. Therefore, adaptive adjustments of a_i^2 enable the proposed model relatively flexible switch between the prototype-based and the surface-based classifiers.

2.2. LEARNING ALGORITHM

Now we set the weights $v_{j(d+1)}(j = 1, \ldots, N_h)$ between the extra node in the input layer and the j-th hidden node in the ICBP to all $+1$ or all -1 respectively and call its corresponding ICBP as ICBP+1 or ICBP$-$1. By o_i $(i = 1, \ldots, N_o)$ denote the network expected outputs and a corresponding sum-of-squares error function is defined as:

$$E = \frac{1}{2}\sum_i (o_i - y_i)^2 \tag{3}$$

Adopting the known-as error back-propagation (EBP) learning algorithm (in fact, any other improved algorithms can be applied), the weight adjustment quantities between the output and hidden layers are easily derived as follows:

$$\Delta w_{ij}(t) = -\eta \frac{\partial E_p}{\partial w_{ij}} = \eta[o_i(t) - y_i(t)] \cdot h_j(t), \quad i = 1, \ldots, N_o, \ j = 0, \ldots, N_h \tag{4}$$

And the adjusting quantities in the weights between the hidden and input layers are:

$$\Delta v_{jk}(t) = \eta \left[\sum_{l=1}^{N_o} (o_l(t) - y_l(t)) w_{lj} \right] \cdot h_j(t) \cdot (1 - h_j(t)) x_k(t) \ j = 1, \ldots, N_h, k = 0, \ldots, d \tag{5}$$

Finally corresponding formula for $\Delta a_k, (k = 1, \ldots, d)$ are:

$$\Delta a_k = 2\eta \sum_{j=1}^{N_h} \left\{ \sum_{i=1}^{N_o} \left[(o_i - y_i) w_{ij} \right] \cdot h_j (1 - h_j) v_{j,d+1} \right\} \cdot a_k x_k^2, \quad k = 1, \ldots, d \tag{6}$$

2.3. PARTIALLY OVERCOME THE LOCAL MINIMA IN CBP AND ICBP

Like the other network models adopting gradient descent learning algorithm, CBP and ICBP also suffer from local minima and slow convergence. To speedup convergence, some researchers have put forward fast propagation and brute force learning algorithm. While in order to jump out of the local minima, we can choose simulation annealing (SA) or other related improving algorithms. Their common ground is to

add controlled variance noise to the weight vectors so as to increase the opportunities of jumping out of the local minima. At the same time, the noise weakens gradually to make the algorithm convergent to a global minimum. Theoretically speaking, SA can overcome the problem of local minima. But its convergent speed is still slow. And due to the probability characteristic of SA weights updating, it cannot realize online learning efficiently and needs to control abundant internal weight noises. So if adjusting the external input, desired output and learning rate instead, we may possibly realize effective online learning. Therefore, we add white noise to the desired outputs of CBP and ICBP and properly adjust the noise variance and amplitude to increase the chance of overcoming the local minima, at the same time converging to the result of the original optimization problem [3, 7]. In the following we will take the network structure shown in Figure 1 as an example to analyze the influence to weight adjustments after adding noise to the desired outputs. Consider the following desired objective function:

$$J = E\left[\frac{1}{2}\sum_{i=1}^{N_o} e_i(t)^2\right] = E\left[\frac{1}{2}\sum_{i=1}^{N_o}(o_i(t) - y_i(t))^2\right] \tag{7}$$

Because the sample distribution is unknown, in practice we use directly the sum of squared error instead:

$$\varepsilon(t) = \frac{1}{2}\sum_{i=1}^{N_o}(e_i(t))^2 \tag{8}$$

Now we introduce the normally distributed white noise $n_i(t)$, with the zero mean and the variance, σ^2, to the desired outputs $o_i(t)$. Suppose the noise is independent from the inputs $x_k(t)$ and the desired outputs $o_i(t)$. Thus the new desired outputs become:

$$o_i'(t) = o_i(t) + n_i(t), \quad (i = 1, \ldots, N_o) \tag{9}$$

The function (7) can now be simplified as:

$$\begin{aligned}
J &= E\left[\frac{1}{2}\sum_{i=1}^{N_o}(o_i'(t) - y_i(t))^2\right] = \frac{1}{2}E\left[\sum_{i=1}^{N_o}(o_i(t) + n_i(t) - y_i(t))^2\right] \\
&= \frac{1}{2}E\left\{\sum_{i=1}^{N_o}[y_i(t) - E((o_i(t) + n_i(t))/x(t))]^2\right\} + \\
&\quad + \frac{1}{2}E\left\{\sum_{i=1}^{N_o}[o_i(t) + n_i(t) - E((o_i(t) + n_i(t))/x(t))]^2\right\} - \\
&\quad - E\left\{\sum_{i=1}^{N_o}[y_i(t) - E((o_i(t) + n_i(t))/x(t))] \cdot [o_i(t) + n_i(t) - E((o_i(t) + n_i(t))/x(t))]\right\} \\
&= \frac{1}{2}E\left\{\sum_{i=1}^{N_o}[y_i(t) - E((o_i(t) + n_i(t))/x(t))]^2\right\} + \frac{1}{2}E\left\{\sum_{i=1}^{N_o}\mathrm{Var}[(o_i(t) + n_i(t))/x(t)]\right\}
\end{aligned} \tag{10}$$

When adopting the EBP algorithm, only the first item in Equation (10) produces influence on weight adjustments. From the fact that $n_i(t)$ is independent from $o_i(t)$ and $x_k(t)$, $(k = 0, \ldots, d+1)$ and $E[n_i(t)] = 0$, we have:

$$E\{(o_i(t) + n_i(t))/x(t)\} = E\{o_i(t)/x(t)\} \tag{11}$$

The above equation indicates that in the statistical sense, the optimization with adding noise to the desired output is equivalent to that without doing. In other words, noise addition will not produce any ill-posed results. Hence it will not influence the learning algorithm.

$$\begin{aligned}\varepsilon_{\text{noise}}(t) &= \frac{1}{2}\sum_{i=1}^{N_o}[o_i(t) + n_i(t) - y_i(t)]^2 = \frac{1}{2}\sum_{i=1}^{N_o}[e_i(t) + n_i(t)]^2 \\ &= \varepsilon(t) + \frac{1}{2}\sum_{i=1}^{N_o} n_i^2(t) + \sum_{i=1}^{N_o} e_i(t)n_i(t)\end{aligned} \tag{12}$$

Corresponding to Equations (4), (5) and (6), the ICBP weights are adjusted as follows:

$$\Delta w_{ij}(t) = -\eta\frac{\partial}{\partial w_{ij}}\varepsilon(t) + \eta n_i(t)h_j(t), \quad i = 1, \ldots, N_o, \;\; j = 0, \ldots, N_h \tag{13}$$

$$\begin{aligned}\Delta v_{jk}(t) = &-\eta\frac{\partial}{\partial v_{jk}}\varepsilon(t) + \\ &+ \eta\sum_{i=1}^{N_o}[n_i(t)w_{ij}(t)]\cdot h_j(t)(1 - h_j(t))x_k(t) \quad j = 1, \ldots, N_h, \;\; k = 0, \ldots, d\end{aligned} \tag{14}$$

$$\begin{aligned}\Delta a_k(t) = -\eta\frac{\partial}{\partial a_k}\varepsilon(t) + 2\eta\sum_{j=1}^{N_h}\Bigg\{\sum_{i=1}^{N_o}[n_i(t)w_{ij}(t)]\cdot h_j(t)\times \\ \times(1 - h_j(t))v_{j(d+1)}(t)\Bigg\}\cdot a_k(t)x_k^2(t) \quad k = 1, \ldots, d\end{aligned} \tag{15}$$

For details of the above derivation, see Appendix 2.

3. Experimental Results of Time Series Prediction

3.1. CHAOTIC TIME SERIES PREDICTION [8]

Time series produced by iterating the logistic map

$$f(x) = \alpha x(1 - x) \quad \text{(continuous)}$$

or

$$x(n+1) = \alpha x(n)(1 - x(n)) \quad \text{(discrete)}$$

is probably the simplest system capable of displaying deterministic chaos. This first-order difference equation, also known as the Feigenbaum equation, has been extensively studied as a model of biological populations with non-overlapping generations, where $x(n)$ represents the normalized population of n-th generation and α is a parameter that determines the dynamics of the population. The behavior of the time series depends critically on the value of the bifurcation parameter α. If $\alpha < 1$, the map has a single fixed point and the output or population dies away to zero. For $1 < \alpha < 3$, the fixed point at zero becomes unstable and a new stable fixed point appears. So the output converges to a single nonzero value. As the value of α increases beyond three, the output begins to oscillate first between two values, then four values, then eight values, and so on, until α reaches a value of about 3.56 when the output becomes chaotic. In this contrastive experiment, we set $\alpha = 3.56$ and produce 100 elements sequence orderly. First, we use the training data pairs of $(x_t, x_{t+1}), (1 \leqslant t < 99)$ to do 99 times of trainings, and then we take 100 data points equally spaced in [0, 0.99] as the test data to do 100 times of single step predictions. The experimental results are averaged from 50 times of experiments. Noise is added in the experiments of all other network models except RBF. The experiment results show that the best forecasting performance is obtained when the number of hidden nodes equals 3 to 6. Figure 2 shows the contrastive experimental results of BP, CBP, RBF, ICBP−1 and ICBP+1 when $N_h = 4$. In this experiment, RBF network behaves the best, and ICBP−1 is inferior to it. MVAR, the 50 times of average of the sum of squared difference between the 100 predicting results and targets, is chosen as the performance measure as shown in Table 1.

$$\mathrm{MVAR} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{P}\sum_{k=1}^{N_o}[o_i(t) - y_i(t)]^2,$$

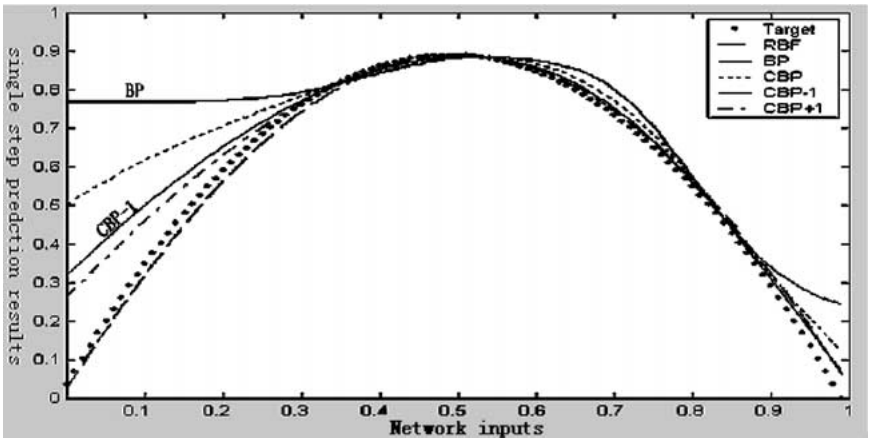

Figure 2. Chaotic time series single step prediction results on [0, 0.99], compared with the targets, of the respective network model.

Table 1. The experiment results of single step chaotic time series prediction. The table items are 50 times of averages of the sum of the squared difference between the predicting results, of the 100 data points equally spaced in [0, 0.99], and the targets. The measure LP in this and all the following tables represents the discrepancy of the number of adaptable parameters of ICBP and CBP.

Hidden nodes number	LP	BP	CBP	ICBP − 1	ICBP + 1	RBF
1	0	7.7804	5.8576	8.4088	6.9654	
2	1	4.9787	3.4984	3.8145	3.5984	
3	2	5.1838	3.9128	3.3810	2.6582	
4	3	5.3162	6.1919	3.3489	2.6680	
5	4	5.8911	3.5253	3.9276	3.5803	0.0491
6	5	6.2600	4.0440	4.1350	2.6568	RBF needs
7	6	7.2787	4.1152	3.3827	3.8760	two radial
8	7	7.7271	5.1853	4.0377	5.4475	basis neurons
9	8	9.8909	5.3998	5.1877	3.8164	
10	9	11.5356	4.3536	4.4770	5.0264	

where N represents N times of repeated experiments; P represents P predicting points; N_o is the number of output nodes. Here $N = 50$, $P = 100$, $N_o = 1$.

3.2. MULTIPLE INPUTS AND MULTIPLE OUTPUTS PREDICTION [8]

We also do experiments in an MIMO nonlinear system. The two-dimensional input-output vectors of the plant were assumed to be $X(t) = [x_1(t), x_2(t)]^T$ and $Y(t) = [y_1(t), y_2(t)]^T$. The difference equation describing the plant was assumed to be of the form

$$\begin{bmatrix} y_{p1}(t+1) \\ y_{p2}(t+1) \end{bmatrix} = \begin{bmatrix} f_1\lfloor y_{p1}(t), y_{p2}(t), x_1(t), x_2(t) \rfloor \\ f_2[y_{p1}(t), y_{p2}(t), x_1(t), x_2(t)] \end{bmatrix}$$

where the known function f_1 and f_2 have the forms

$$f_1(y_{p1}, y_{p2}, x_1, x_2) = \frac{0.8y_{p1}^3 + x_1^2 x_2}{2 + y_{p2}^2}$$

and

$$f_2(y_{p1}, y_{p2}, x_1, x_2) = \frac{y_{p1} - y_{p1}y_{p2} + (x_1 - 0.5)(x_2 + 0.8)}{1 + y_{p2}^2}$$

In this experiment $x_1(t) = \sin(2\pi t/250)$ and $x_2(t) = \cos(2\pi t/250)$. We produce 200 steps of sequence from the above equation. After processing the pth ($1 \leqslant p \leqslant 100$) training using the training data pairs of $(\vec{x}_t, \vec{y}_{t+1})_p$, $(1 \leqslant t < 98 + p)$, we take x_{100+p} ($1 \leqslant p \leqslant 100$) as the test data to do 100 times of single step prediction. Noise

is added in the experiments of all other network models except RBF. The experiments are repeated for 50 times and averaged results are obtained as shown in Table 2. The items in Table 2 are MVAR(here $N = 50$, $P = 100$, $N_o = 2$). The models exhibit relatively good predicting performance when their hidden layers take 2 to 6 nodes. Figure 3 and 4 show the predicting results of RBF, CBP and ICBP−1 compared with the original time series. In this experiment, RBF behaves rather bad while ICBP−1 exhibits relatively excellent performance.

3.3. APPLICATIONS TO CITY DAILY LIFE WATER CONSUMED QUANTITY

Predicting city daily life water consumed quantity, especially multiple-steps, can help to lay a productive course, economize energy sources and boost production benefits. It is practically valuable for civil life and manufacture. In terms of the historical data provided, we predict the water consumed quantity one month to one quarter ahead for the use of water supply department. There exist two methods to predict by neural networks: static method (predicting the next n steps of values in a single running) and dynamic method (in one epoch, only forecasting the values in succession to the inputs, and taking these values as the inputs to process the next prediction). Since the static method optimize aiming at long-term and short-term errors simultaneously, it is relatively difficult for this method to obtain short-term optimized predicting result. According to the practical condition of this experiment, we choose the dynamic prediction way. Generally, the dynamic predicting model can be described as follows:

$$y_{i,t} = F[W_{i,t}, y_{i,t-1}, \ldots, y_{i,t-m}, y_{i-1,t}, \ldots, y_{i-1,t-m}, \ldots, y_{i-n,t}, \ldots, y_{i-n,t-m}],$$

where the first subscript represents the year and the second the month. That is to say, the information of the preceding m months, the month of the preceding n years and the preceding m months of the preceding n years are taken into the network inputs in prediction. Not supplied with enough data, we take n as zero here. In the practical experiment we process single step prediction for the recent twelve months, employing networks with different structural parameters separately. The network inputs include the year, the month, the planned consuming quantity of the month and the actual consumed quantity of the preceding m months. Because the values of the monthly planned consuming quantity and the monthly actual consumed quantity are huge, they are mapped into the district of (0, 1). Let M be the set of the monthly planned consuming quantity or the monthly actual consumed quantity; let IN_i be the ith monthly planned consuming quantity or the ith monthly actual consumed quantity and in_i be the ith actual input to the network; then

$$in_i = \frac{IN_i - \min(M)}{\max(M) - \min(M)} 0.8 + 0.1$$

Table 3 shows the experiment results contrastively in the condition of different network structural parameters and different network inputs, where $P = 1$ represents

Table 2. The experiment results of two dimensional time series single step prediction. Table items are the 50 times of averages of the sum of squared differences between the predicting results and the targets of the latter 100 data points.

		The first dimension					The second dimension				
N^*	LP	BP	CBP	ICBP − 1	ICBP + 1	RBF	BP	CBP	ICBP − 1	ICBP + 1	RBF
1	−1	0.8199	0.8252	0.8444	0.8339		0.2839	0.2832	0.2375	0.2437	5.8669
2	0	0.0522	0.1312	0.0180	0.0465		0.3026	0.6382	0.1335	0.1486	
3	1	0.0214	0.0426	0.0145	0.0703		0.2214	0.2316	0.1694	0.2239	
4	2	0.0315	0.0372	0.0159	0.0218		0.3074	0.2624	0.2300	0.2231	
5	3	0.0374	0.0275	0.0194	0.0182		0.3014	0.2673	0.2323	0.2046	
6	4	0.0378	0.0361	0.0156	0.0216	0.3211 RBF needs	0.3498	0.2651	0.2662	0.2140	
7	5	0.0550	0.0331	0.0285	0.0194	17 radial basis neurons	0.3637	0.2914	0.2704	0.2245	
8	6	0.0491	0.0428	0.0185	0.0245		0.3544	0.2988	0.2334	0.2348	
9	7	0.0578	0.0467	0.0303	0.0228		0.4063	0.3943	0.3148	0.2785	
10	8	0.0849	0.0709	0.0329	0.0367		0.4864	0.3159	0.2833	0.2503	
11	9	0.0852	0.0573	0.0348	0.0361		0.5038	0.2848	0.3357	0.2463	
12	10	0.1095	0.1024	0.0411	0.0362		0.5131	0.3569	0.3500	0.2791	

N^*: The number of the hidden nodes.

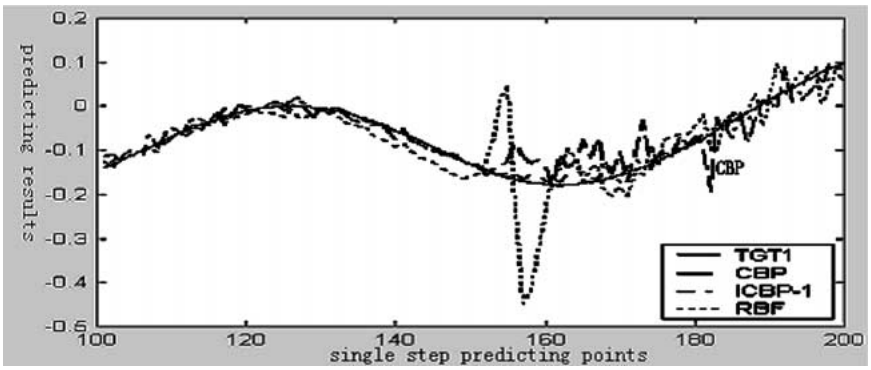

Figure 3. The first dimensional single step predicting results of the latter 100 data points by CBP, ICBP−1 and RBF in comparison with the targets.

adding planned consuming quantity of the month, whereas $P = 0$ denotes the opposite meaning; and $M = 0, \ldots, 5$ represents adding the actual consumed quantity of the preceding M months. MPE is taken as the performance measure for this experiment, which is defined as follows:

$$\mathrm{MPE} = \frac{1}{\mathrm{NPN}_o} \sum_{i=1}^{N} \sum_{j=1}^{P} \sum_{k=1}^{N_o} \left| \frac{\mathrm{Prediction}_k^j - \mathrm{Actual}_k^j}{\mathrm{Actual}_k^j} \right|,$$

where P represents P steps of predicting points, N represents N times of repeated experiments and N_o is the number of output nodes. Here we set $N = 50$, $P = 12$, $N_o = 1$. The experiment results indicate that the predicting effects are the best when adding to the network inputs the information of the planned consuming quantity of the month and the actual consumed quantities of the preceding two to three months. In this experiment the RBF network needs 26 to 29 radial basis neurons, however, its

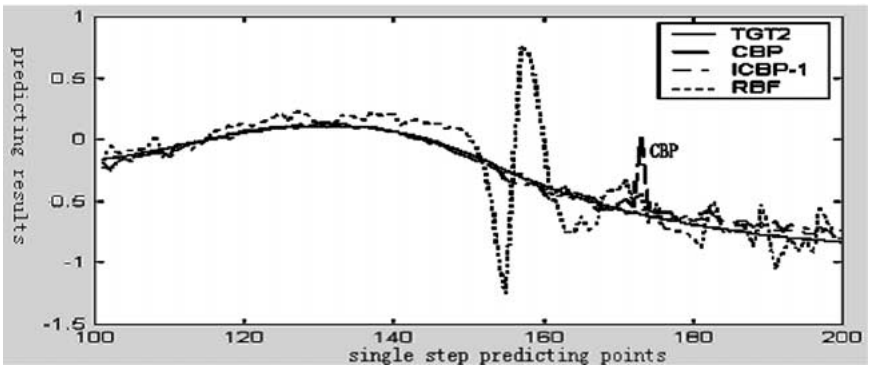

Figure 4. The second dimensional single step predicting results of the latter 100 data points by CBP, ICBP−1 and RBF in comparison with the targets.

Table 3. The experiment results of water consumed quantity single step prediction. Table items are the 50 times of averages of the measure MPE of the predicting results of the recent 12 months. The RBF network needs 26–29 radial basis neurons.

Network structure		LP	BP	CBP	ICBP − 1	ICBP + 1	RBF
$P = 0$, $M = 0$	2-4-1	2	10.56	9.07	10.41	9.40	12.51
$P = 0$, $M = 1$	3-6-1	3	9.59	9.69	8.72	10.01	10.93
	4-6-1	2	6.34	5.85	6.11	5.06	
$P = 0$, $M = 2$	4-10-1	6	6.89	6.30	5.85	5.41	10.35
	4-12-1	8	8.90	7.56	5.87	7.24	
	5-6-1	1	6.17	5.89	5.92	5.95	
$P = 0$, $M = 3$	5-9-1	4	5.65	6.99	5.67	5.00	9.40
	5-11-1	6	6.58	7.19	6.80	5.24	
$P = 0$, $M = 4$	6-8-1	2	7.97	7.81	10.19	7.25	8.43
$P = 0$, $M = 5$	7-10-1	3	8.52	7.73	9.79	9.57	11.21
$P = 1$, $M = 0$	3-6-1	3	11.43	11.60	9.34	8.28	10.44
$P = 1$, $M = 1$	4-6-1	2	8.13	8.07	5.85	6.76	8.76
	5-6-1	1	5.02	6.20	4.94	5.25	
$P = 1$, $M = 2$	5-8-1	3	5.14	5.03	3.39	4.90	8.04
	5-11-1	6	7.42	6.64	5.61	4.61	
	6-7-1	1	6.65	5.86	5.16	5.57	
$P = 1$, $M = 3$	6-10-1	4	6.42	5.14	5.82	5.24	6.80
	6-15-1	9	7.42	6.99	6.16	3.66	
$P = 1$, $M = 4$	7-10-1	3	5.95	6.11	6.52	5.33	8.54
$P = 1$, $M = 5$	8-12-1	4	10.15	9.65	10.21	7.68	8.98

predicting performance are still poor. Figure 5 shows the single step prediction results of the recent twelve months employing RBF, CBP and ICBP−1 in comparison with the actual consumed quantities.

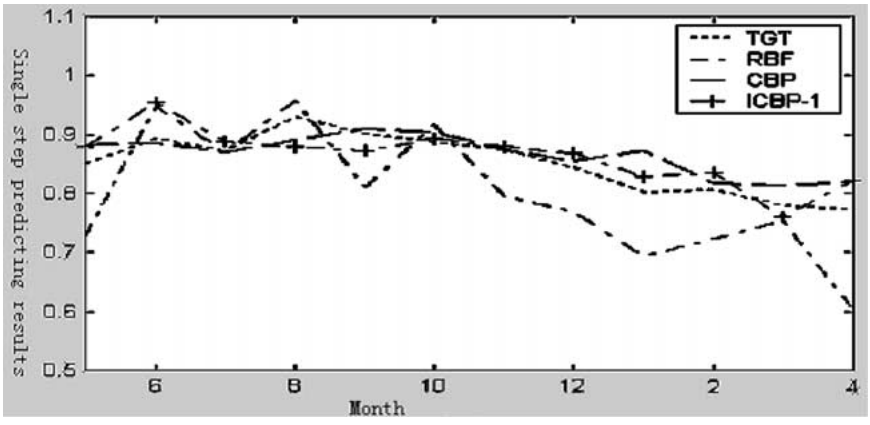

Figure 5. The single step prediction results of the recent 12 months water consumed quantity, compared with the actual consumed quantity, by RBF, CBP and ICBP−1.

4. Conclusion

Retaining the original structure of CBP, we obtain ICBP network model through a special construction to the extra node in CBP input layer, which still adopt the regular BP algorithm. Adding noise to the desired outputs in training process conduces to partially overcome the local minima. Furthermore, the special assignments to the weights between the node and the hidden layer reduce the adaptable parameters of CBP. On the one hand, this can bring learning advantages; on the other hand, different assignments can realize more flexible classification (i.e. can construct ellipsoidal or hyperboloid separation surface directly). Generally, the MPE measure of ICBP is much better than BP and CBP, whereas the computational complexity of ICBP does not increase. The experiment results show that the performance of CBP is only inferior to RBF in the single-input and single-output time series prediction. ICBP performs the best in the multiple-input and multiple-output nonlinear time series prediction, while the performance of RBF is poor. In the predictions of water consumed quantity, the MPE of ICBP can be lower than 5% through properly adjusting the inputs and the number of hidden nodes, which can fully satisfy the needs of this type of application. Although RBF possesses faster training speed than ICBP, however, this type of application has low demands to training speed, so ICBP meets the application demands better. In addition, is it useful for improving the prediction performance to add to the inputs the difference information reflecting the changing trend? In the prediction of complex changing variables, is it conducive for reducing the number of hidden nodes and boosting the forecasting capability to connect the input layer and the output layer directly? All these problems need our further study.

Appendix 1. Equivalence to RBF

When $v_{jd+1} < 0$, Equation (2) can be rewritten as:

$$r_j(\vec{x}, \vec{v}) = -d_j^2 - \theta'_j, \quad \text{where } d_j^2 = |v_{jd+1}|(\vec{x} - \vec{c}_j)\Lambda(\vec{x} - \vec{c}_j)^T, \theta' = \theta_j / v_{d+1}$$

Due to $\sigma(r_j) = \frac{1}{1+e^{-r_j}} = \frac{1}{1+e^{d_j^2+\theta'_j}} = \frac{e^{-d_j^2}}{e^{-d_j^2}+e^{\theta'_j}}$, we multiply $e^{\theta'_j}$ to the two sides of the formula. When $e^{\theta'_j}$ becomes big enough, the output will approach $e^{-d_j^2}$, which will realize an approximation to RBF network.

Appendix 2. The Influence of Noise Addition on Weight Adjustments

Now, let's observe the influence extent on the weight adjustments. It can be known from Equations (13), (14) and (15) that the influence on the weight adjustments all come from the second items.

As to $\Delta w_{ij}(t)$, because

$$E[\eta n_i(t) h_j(t)] = \eta E[n_i(t)] E[h_j(t)] = 0 \tag{A1}$$

and

$$\mathrm{Var}[\eta n_i(t)h_j(t)] = \eta^2\sigma^2\,\mathrm{Var}[h_j(t)] \tag{A2}$$

so that:

$$\mathrm{Var}(\Delta w_{ij}(t)) \propto \eta^2\sigma^2\,\mathrm{Var}(h_j(t))$$

As to $\Delta v_{jh}(t)$, because

$$E\left\{\eta\sum_{i=1}^{N_o}[n_l(t)w_{lj}(t)]\cdot h_j(t)(1-h_j(t))x_k(t)\right\}$$
$$=\sum_{i=1}^{N_o}\{E[n_l(t)]E[\eta w_{lj}(t)\cdot h_j(t)(1-h_j(t))x_k(t)]\}=0 \tag{A3}$$

$$\mathrm{Var}\left\{\eta\sum_{l=1}^{N_o}[n_l(t)W_{lj}(t)]\cdot h_j(t)(1-h_j(t))x_k(t)\right\}$$
$$=\eta^2\sigma^2\mathrm{Var}\left\{\left[\sum_{l=1}^{N_o}W_{lj}(t)\right]\cdot h_j(t)(1-h_j(t))x_k(t)\right\} \tag{A4}$$

so that:

$$\mathrm{Var}(\Delta v_{jk}(t)) \propto \eta^2\sigma^2\,\mathrm{Var}\left\{\left[\sum_{l=1}^{N_0}W_{lj}(t)\right]\cdot h_j(t)(1-h_j(t))x_k(t)\right\}$$

Because:

$$E\left[2\eta\sum_{j=1}^{N_h}\left\{\sum_{i=1}^{N_o}[n_i(t)w_{ij}(t)]\cdot h_j(t)(1-h_j(t))v_{j,d+1}(t)\right\}\cdot a_k(t)x_k^2(t)\right]$$
$$=2\eta a_k(t)x_k^2(t)\cdot\sum_{i=1}^{N_o}\left\{E[n_i(t)]\cdot E\left[\sum_{j=1}^{N_h}w_{ij}(t)h_j(1-h_j(t))v_{j,d+1}\right]\right\}=0 \tag{A5}$$

$$\mathrm{Var}\left[2\eta\sum_{j=1}^{N_h}\left\{\sum_{i=1}^{N_o}[n_i(t)w_{ij}(t)]\cdot h_j(t)(1-h_j(t))v_{j,d+1}(t)\right\}\cdot a_k(t)x_k^2(t)\right]$$
$$=4\eta^2\sigma^2\cdot\mathrm{Var}\left[\sum_{i=1}^{N_o}\sum_{j=1}^{N_h}[w_{ij}(t)h_j(1-h_j(t))v_{j,d+1}]\cdot a_k(t)x_k^2(t)\right] \tag{A6}$$

$$\mathrm{Var}(\Delta a_k(t)) \propto 4\eta^2\sigma^2\cdot\mathrm{Var}\left[\sum_{i=1}^{N_0}\sum_{j=1}^{N_h}[w_{ij}(t)h_j(1-h_j(t))v_{jd+1}]\cdot a_k(t)x_k^2(t)\right]$$

Thereby, it can be known from Equations (A2), (A4) and (A6) that when $\eta^2\sigma^2 \to 0$, the noise influences to the weight adjustments all tend towards zero. Hence this method is proved to be effective. In the practical implementation of the algorithm,

we make σ^2 rather big in the beginning to enable it jump out of the local minima. Then during the learning process, we let $\eta^2\sigma^2$ go to zero gradually, so that the original optimization results can be obtained. The experiment results show that the method is useful for overcoming local minima and expediting the convergent speed.

Acknowledgements

Supported by Natural Science foundation (grant No.BK2002092) and 'QingLan' project of JiangSu province and Returnee foundation of China.

References

1. Ridella, S., Rovetta, S. and Zunino, R.: Circular backpropagation networks for classification, *IEEE Transactions on Neural Networks* **8**(1) (1997), 84–97.
2. Ridella, S., Rovetta, S. and Zunino, R.: Circular backpropagation networks embed vectior quantization, *IEEE Transactions on Neural Networks* **10**(4) (1999), 972–975.
3. Gorp, J. V., Schoukens, J. and Piktelon, R.: Learning neural networks with noisy inputs using the errors-in-variables approach, *IEEE Transactions on Neural Networks* **11**(2) (2000), 402–413.
4. Benzhu, Z. and Songcan, C.: Equivalence between vector quantization and ICBP networks, *Journal of Data Acquisition and Processing (in Chinese)* **16**(3) (2001), 291–294.
5. Benzhu, Z.: The research on the performance and applications of improved BP neural networks, Thesis of Master degree, Feb. 2001, Nanjing University of Aeronautics and Astronautics.
6. Benzhu, Z. and Songcan, C.: The equivalence between ICBP and the Bayesian classifier, Tech. Report No. 021, 2001, Dept. of Computer Science & Engineering, Nanjing University of Aeronautics and Astronautics.
7. Netlab toolbox available at http://www.ncrg.aston.ac.uk/
8. Philip Chen, C. L. and Wan, J. Z.: A Rapid learning and dynamic stepwise updating Algorithm for flat neural networks and the application to time-series prediction, *IEEE Transaction on Syst*., Man & Cyberne.-part B:Cybern. **29**(1) (1999), 62–72.