Neurocomputing 137 (2014) 47–56

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Time series forecasting using a deep belief network with restricted Boltzmann machines

Takashi Kuremoto [a,*,1], Shinsuke Kimura [a], Kunikazu Kobayashi [b], Masanao Obayashi [a]

[a] Graduate School of Science and Engineering, Yamaguchi University, Tokiwadai 2-16-1, Ube, Yamaguchi 755-8611, Japan
[b] School of Information Science & Technology, Aichi Prefectural University, Ibaragabasama 1522-3, Nagakute, Aichi 480-1198, Japan

ARTICLE INFO

Article history:
Received 5 December 2012
Received in revised form
13 March 2013
Accepted 29 March 2013
Available online 22 January 2014

Keywords:
Time series forecasting
Deep belief nets
Restricted Boltzmann machine
Multi-layer perceptron
CATS benchmark
Chaos

ABSTRACT

Multi-layer perceptron (MLP) and other artificial neural networks (ANNs) have been widely applied to time series forecasting since 1980s. However, for some problems such as initialization and local optima existing in applications, the improvement of ANNs is, and still will be the most interesting study for not only time series forecasting but also other intelligent computing fields. In this study, we propose a method for time series prediction using Hinton and Salakhutdinov's deep belief nets (DBN) which are probabilistic generative neural network composed by multiple layers of restricted Boltzmann machine (RBM). We use a 3-layer deep network of RBMs to capture the feature of input space of time series data, and after pretraining of RBMs using their energy functions, gradient descent training, i.e., back-propagation learning algorithm is used for fine-tuning connection weights between “visible layers” and “hidden layers” of RBMs. To decide the sizes of neural networks and the learning rates, Kennedy and Eberhart's particle swarm optimization (PSO) is adopted during the training processes. Furthermore, “trend removal”, a preprocessing to the original data, is also approached in the forecasting experiment using CATS benchmark data. Additionally, approximating and short-term prediction of chaotic time series such as Lorenz chaos and logistic map were also applied by the proposed method.

1. Introduction

Artificial neural networks (ANNs) have been popularly used for time series forecasting since 1980s. As indicated by Crone and Nikolopoulos [1], till 2007, there have been more than 5,000 publications on ANNs for forecasting. The well-known ANNs of predictors can be listed as multi-layer perceptron (MLP), recurrent neural networks (RNN), radial basis function networks (RBFN), and many varieties of them. To train these models, i.e., finding the suitable parameters of them, gradient descent methods are often used. The most popular learning algorithm among these methods is Rumelhart, Hinton and Williams's “error back-propagation” (BP) [2] which can be regarded as “a mile-stone” in the field of ANN research.

However, there are still problems when ANNs are applied to real world problems:

(i) How to decide the structure of them? It has been proved that a MLP with 1 hidden layer and enough units of the hidden layers can realize approximation of arbitrary nonlinear functions [3]. However, it is also known that the larger size of for the hidden layer yields the “over fitting” problem, adaptive structure of MLP is necessary for real problems [4,5].

(ii) How to decide the initial values of connection weights? BP is a supervised learning which uses teacher signals (samples of input and output data) to modify weights of connections (synapses) between units (neurons) on the different layers, so the initial values of these connections affect the learning process, which is a kind of nonlinear mapping of input space to output space. Suitable initial weights may accelerate the learning convergence and avoiding the learning process to stop at local optima.

(iii) How to find a suitable learning rate during training? Learning rate, i.e., the rate of parameters modification within one step of training, also affects the speed of learning process. Too high learning rate may result a unstable learning convergence, and small learning rate resists the learning process.

To solve the first problem, Kuremoto et al. proposed a self-organizing fuzzy neural network (SOFNN) as a time series predictor which units were generated by the input data [6–8].

For the second problem, Hinton et al. proposed a deep belief nets (DBN) [9] which uses a layer-by-layer unsupervised learning to pre-training the initial weights of the networks, and then with

[*] Corresponding author. Tel: +81 836 85 9520, fax: +81 836 85 9501.
E-mail addresses: wu@yamaguchi-u.ac.jp (T. Kuremoto), kobayashi@ist.aichi-pu.ac.jp (K. Kobayashi), m.obayas@yamaguchi-u.ac.jp (M. Obayashi).

[1] A part of this work was supported by Grant-in-Aid for Scientific Research (JSPS 23500181).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.03.047

global supervised learning for fine-tuning. DBN has been successfully applied to dimensionality reduction (image compression) [10], digit recognition [11], acoustic representation [12], and it is still developed ongoing [13].

In this paper, we propose a 3-layer DBN which is composed by two restricted Boltzmann machines (RBMs) [14,15] to be a predictor for time series forecasting. In fact, according to [9–13], when the high dimension data are input to the units of "visible layer" of a RBM, then the units of "hidden layer" of RBM detects the feature of data between different classes according to the connection weights. The connection of units of RBM is restricted to different layers, i.e., no connection exists between the units of a same layer, so the paired layers are named "restricted Boltzmann machine" (RBM). Meanwhile, when the hidden layer is used as another visible layer, a new hidden layer estimates the feature of the classified data, i.e., the second RBM estimates "the feature of features" of the input data. Using these combined RBMS, which is named a deep belief network (DBN), we construct a time series approximation model and apply it to be a predictor. The adjustment of connectional weights of RMBs in the DBN is given by the descent of probabilities of energy functions of visible layers and hidden layers at first, and fine-tuning using BP [2] is adopted after the unsupervised learning. To decide the number units of input layer and hidden layer, and suitable learning rate, we adopt particle swarm optimization (PSO) [16] into the design of DBN. Additionally, to meet the characteristic of neural network prediction models [17,18], seasonal factors (trends) in the original data are removed, i.e., preprocessed "difference time series" is adopted for the proposed predictor.

To verify the effectiveness of our prediction model, CATS benchmark was used whereas it served as the problem of time series prediction competition since 2004 [19,20]. Furthermore, time series data of Lorenz chaos [23] were also investigated as a typical determinant nonlinear phenomenon. The prediction results showed that the proposed DBN with RBMs gave higher forecasting precision comparing with MLP [2] and traditional prediction model ARIMA [21].

2. A predictor using DBN of RBMs

A predictor using DBN of RBMs is proposed and the learning algorithm is given in Section 2.1. Structure and parameter optimization using PSO will be stated in Section 2.2, and preprocessing to the original time series data is in Section 2.3.

2.1. Architecture and learning algorithms

Artificial models with deep architectures (DAs) have become a kind of powerful tools to retrieve or represent the high-level abstract features of high dimension data [9–12]. In this paper, we propose a deep belief net (DBN) with two RBMs as shown in Fig. 1 to be a model of time series prediction. The model training process is as follows.

Step 1. Initialization.

Step 1.1 All binary units on each layer of RBMs are set as 0 or 1 randomly, and symmetrical connection weights between units of different layers w_{ij} with random value in (0, 1). Units of the same layer of RBMs do not connect to each other.

Step 1.2 Input space reconstruction. For time series data $x(t), t = 1, 2, ..., T$, reconstructed time series data $x(t-\tau), x(t-2\tau), ..., x(t-n\tau)$ are prepared as the input to the units v_i of visible layer of RBM1, where τ is a positive integer, $i = 1, 2, ..., n$, n is the dimension of the input space, i.e., the number of units of visible layer of RBM1 and $if\, t-\tau \le 0, x(t-\tau) = x(t); t-n\tau \le T$. The value of units on visible layer of RBM1 then equal to the input data: $v_1 = x(t-\tau), v_2 = x(t-2\tau), ..., v_n = x(t-n\tau)$.

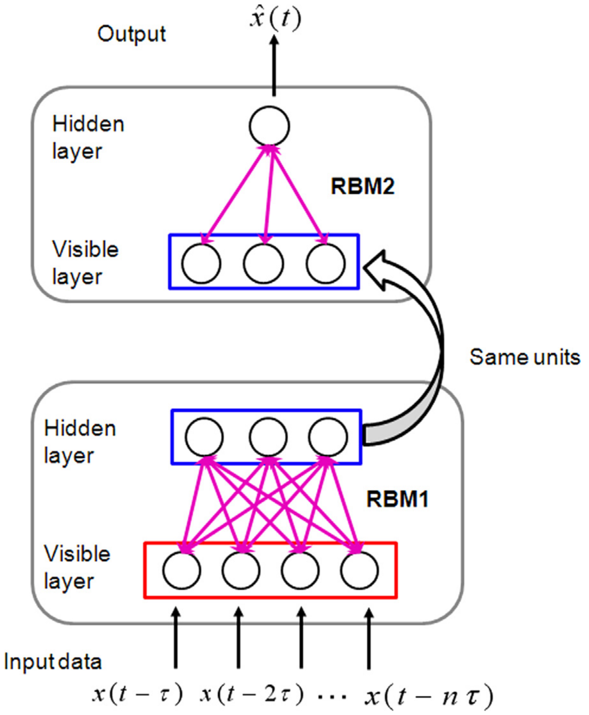

Fig. 1. A predictor constructed by a deep belief network (DBN) with two restricted Boltzmann machines (RBMs).

Step 2. Repeat Step 2.1 to Step 2.5 until energy function E (**v**, **h**) decrease to be a convergent state (here we use a threshold α concerning with the change of E (**v**, **h**) to judge the convergence).

$$E(v,h) = -\textstyle\sum_{i=1}^{n} b_i v_i - \sum_{j=1}^{m} b_j h_j - \sum_{i=1}^{n} \sum_{j=1}^{m} v_i h_j w_{ij} \tag{1}$$

where v_i and h_j are the binary states of input **x**(t) and feature j, b_i and b_j are their biases. Convergence condition can be given as:

$$|A'(v,h) - A(v,h)| < \alpha, \tag{2}$$

$$A(v,h) = \textstyle\sum_{a=k}^{K+k} E_a(v,h)/K, \tag{3a}$$

$$A'(v,h) = \textstyle\sum_{b=k-K}^{k-1} E_b(v,h)/K \tag{3b}$$

where α is a positive parameter and small enough, k is the iteration times and K is a constant indicates the length of evaluation period.

Step 2.1 The value of a unit h_j of hidden layer, i.e., a feature detector, is set to 1 with a probability:

$$p_j = \frac{1}{1+e^{-\Delta E_j}}, \tag{4a}$$

$$\Delta E_j = \textstyle\sum_{i=1}^{n} w_{ij} v_i + b_j \tag{4b}$$

where b_j is the bias of the unit of hidden layer h_j, $j = 1, 2, ..., m$.

Step 2.2 Set v_i to 1 with a probability:

$$p_i = \frac{1}{1+e^{-\Delta E_i}}, \tag{5a}$$

$$\Delta E_i = \textstyle\sum_{j=1}^{m} w_{ij} h_j + b_i \tag{5b}$$

where b_j is the bias of the unit of visible layer v_i, $i = 1, 2, ..., n$.

Step 2.3 Calculate the expectation $p_{ij} = <x(t-i\tau)h_j>_{data}$, where h_j with binary values according to Step 2.1.

Step 2.4 Calculate the expectation $p'_{ij} = <v_i h_j>_{recon}$, where v_i with binary values given by Step 2.2, and h_j from Step 2.1.

Step 2.5 Change the weight w_{ij} as:

$$\Delta w_{ij} = \varepsilon(p_{ij} - p'_{ij}) \quad (6)$$

where ε is a learning rate $(0 < \varepsilon < 1)$.

Step 3. Let the hidden layer of RBM1 to be the visible layer of RBM2 as shown in Fig. 1. Repeat Step 1 and Step 2 for RBM2.
Step 4. The output of the unit of RBM2 $\hat{x}(t)$ is the coarse prediction value of unknown future $x(t)$.
Step 5. Using the difference between $x(t)$ and $\hat{x}(t)$, i.e., their mean squared error (MSE), BP algorithm [2] is executed to fine-tuning weights of each RBM to obtain the refined prediction.
Step 6. Stop training if MSE is small enough, i.e., converged as Eq. (7).

$$\frac{\text{MSE}(L-1) - \text{MSE}(L)}{\text{MSE}(L-1)} < \beta \quad (7)$$

where β is a positive parameter and small enough, L is the iteration time of BP algorithm.
Step 7. Input unlearned time series data to the trained model and predict the future data (one-head, i.e., using $x(t-\tau), x(t-2\tau), ..., x(t-n\tau)$ to output $\hat{x}(t)$, or a long-term prediction, i.e., using $\hat{x}(t), \hat{x}(t+\tau), ..., \hat{x}(t+(n-1)\tau)$ to predict $\hat{x}(t+n\tau)$).

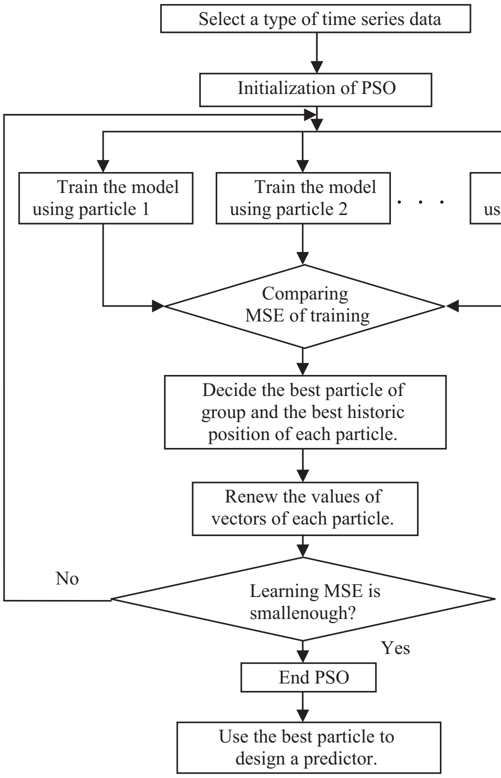
Select a type of time series data
Initialization of PSO
Train the model using particle 1
Train the model using particle 2
. . .
Train the model using particle P
Comparing MSE of training
Decide the best particle of group and the best historic position of each particle.
Renew the values of vectors of each particle.
No
Learning MSE is smallenough?
Yes
End PSO
Use the best particle to design a predictor.

Fig. 2. Using the best particle to design an optimized predictor of DBN.

The learning rule given by Eq. (6) is proved that it works well to train RBM to be an approximate model of the input in [14,15]. And for gradient descent training methods, e.g., BP [2], the initial weights are expected to be set closing to the global minimum, i.e., the optimal solution. So Eq. (6) serves as pretraining before the fine-tuning using the BP learning algorithm, and these learning algorithms make DBN extract abstract features by their multiple output layers.

2.2. Optimization of the model

The structure of a neural network needs to be designed to meet its processing object. The number of layers, the number of units of each layer, learning rate and so on need to be decided when the model is applied to real problems. Here we use a 3-layer DBN, and adopt Kennedy and Eberhart's particle swarm optimization (PSO) [16] to find the optimal numbers of units on input layer and hidden layer, and learning rate of RBM.

Suppose the predictor is given as in Fig. 1, i.e., 2 RBMs are used. The visible layer of RBM1 has n units, the hidden layer of RBM1 has m units as same as the visible layer of RBM2. The hidden layer of RBM2 has 1 unit, i.e., the output of the predictor. Considering the learning rate of RBMε, a particle is designed to be a 3-dimension vector $\mathrm{y}(n, m, \varepsilon)$, where $n = 1, 2, ..., n, m = 1, 2, ..., m, \varepsilon \in (0, 1)$. Preparing population of this kind of particles $\mathbf{y}_i$ with enough size P, i.e. $i = 1, 2, ..., P$, and using the standard algorithm of PSO of [13], the optimized predictor of DBN is able to be constructed. The flow chart of this processing is shown in Fig. 2.

A summarized PSO algorithm used to decide the optimal number of input units and hidden units of DBN is shown as follows:

Step 1. Decide the population size of particles P and limitation of iteration number I.
Step 2. Initialize the start position $\mathrm{y}^{k=0}$ and the start velocity $\mathrm{s}^{k=0}$ of each particle.
Step 3. Evaluate each particle using the mean squared error (MSE) between the prediction value $\hat{x}(t)$ and teacher data $x(t)$, and find the best position of a particle y^k_{ipbest} from its history, and the best particle position of the
swarm y^k_{gbest}.
Step 4. Renew positions and velocities of particles by Eqs. respectively.

$$\mathrm{y}_i^{k+1} = \mathrm{y}_i^k + \mathrm{s}_i^{k+1} \quad (8)$$

$$\mathrm{s}_i^{k+1} = \kappa(\omega \mathrm{s}_i^k + c_1 r_1(\mathrm{y}^k_{ipbest} - \mathrm{y}_i^k) + c_2 r_2(\mathrm{y}^k_{gbest} - \mathrm{y}_i^k)) \quad (9)$$

Where $\kappa, \omega, c_1, c_2, r_1, r_2$ are positive parameters, random values $r_1, r_2 \in (0, 1)$, $i = 1, 2, ..., P$.

Step 5. Finish the algorithm:
if the evaluation function (prediction MSE of training data) converged, or $k = I$;
else $k \leftarrow k+1$
return to Step 3.

2.3. Difference time series

Though artificial neural networks (ANNs) work well for nonlinear data prediction, they reduce learning ability when linear factor exists strongly in the time series. So Zhang proposed to combine the linear prediction model ARIMA [21] with MLP to raise the prediction precision [17]. Meanwhile, Gardner and Mckenzie proposed a smoothing method [18] to preprocessing the original time series data and yield difference time series data $x^d(t)$ as the following:

$$x^d(t) = x(t) - x(t-d) \tag{10}$$

where d is the delay of time, and it is possible to be decided by ARIMA model. An example of this processing result is shown in Fig. 3. The original time series shows the nonlinear with several linear trends, however, a difference time series obtained by a time delay $d=6$ according to Eq. (10) is stabilized around the value 0. The different time series data $x^d(t)$ are used to train ANNs and the ANNs predicted future data will be$\hat{x}^d(t)$. The final prediction result becomes as follows:

$$\hat{x}(t+1) = \hat{x}^d(t+1) + \hat{x}(t-d+1) \tag{11}$$

As the proposed method is a kind of multiple layer ANN, we used Gardner and Mckenzie smoothing method [18] in the case of CATS benchmark, and used Zhang's hybrid models [17], i.e., ARIMA+MLP and ARIMA+SOFNN in the case of chaotic time series forecasting.

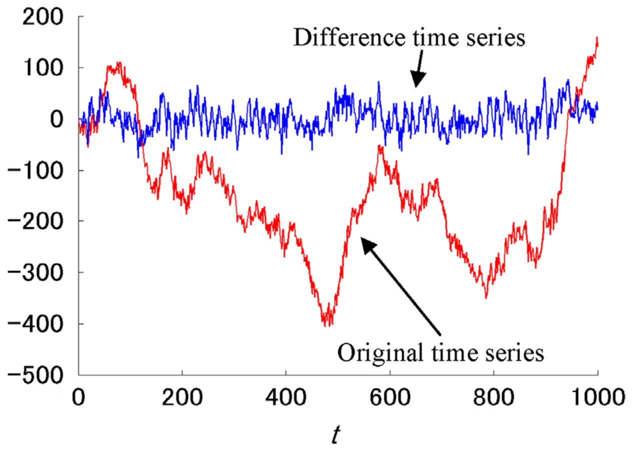

Fig. 3. Original time series data and its difference data obtained by a time delay 6.

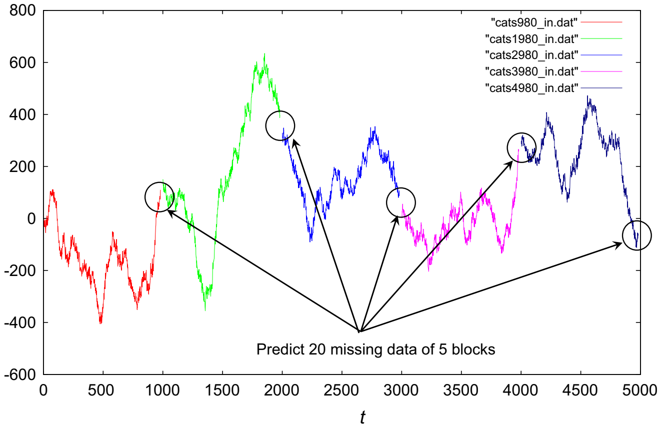

Fig. 4. The artificial time series data of CATS benchmark (IJCNN'04 prediction competition) [19,20].

3. Forecasting result

3.1. The CATS benchmark

To evaluate the performance of the proposed prediction model, the artificial time series data of CATS benchmark [19,20] were used in two prediction experiments.

Experiment I: Long-term prediction.
The CATS data are given in 5 blocks ($t=1, 2, \ldots, 5000$) and in each block, 20 are missing, 980 are known (see Fig. 4). The evaluation functions are the mean square error E_1 and E_2 which computes on the 100 missing values and 80 missing values as follows. Generally, if the internal data between two blocks which time series data are known, then one can use two models corresponding to the former and the latter training samples. However, for Block 5 for CATS benchmark, there is no latter data after the unknown interval. So E_1 may be more strictly to reach higher prediction precision than E_2 according to one's training method.

$$E_1 = \frac{\sum_{t=981}^{1000}(x_t-\hat{x}_t)^2}{100} + \frac{\sum_{t=1981}^{2000}(x_t-\hat{x}_t)^2}{100} + \frac{\sum_{t=2981}^{3000}(x_t-\hat{x}_t)^2}{100} + \frac{\sum_{t=3981}^{4000}(x_t-\hat{x}_t)^2}{100} + \frac{\sum_{t=4981}^{5000}(x_t-\hat{x}_t)^2}{100}, \tag{12}$$

$$E_2 = \frac{\sum_{t=981}^{1000}(x_t-\hat{x}_t)^2}{80} + \frac{\sum_{t=1981}^{2000}(x_t-\hat{x}_t)^2}{80} + \frac{\sum_{t=2981}^{3000}(x_t-\hat{x}_t)^2}{80} + \frac{\sum_{t=3981}^{4000}(x_t-\hat{x}_t)^2}{80} \tag{13}$$

Participants of the forecasting competition in IJCNN'04, gave their long-term prediction results because the missed data were not opened. In Experiment I, 980 data in each block were used to train the models, 20 missed data in each block were predicted where input of the predictors were previous predicted data, i.e., as same as participants of IJCNN'04, long-term prediction was performed.
Experiment II: One-ahead prediction.
We also performed a short-term prediction experiments to

Table 1
Parameters used in the prediction experiments.

Description	Symbol	Quantity
The number of RBM	*RBM1, RBM2*	2
The number of input (visible) layer[a]	$N(1 \le N \le 20)$	Given by PSO
The number of hidden layer[a]	$M(1 \le M \le 20)$	Given by PSO
The number of output	–	1
Interval of input data	τ	1
Learning rate of RBM	ε (Eq. (6))	Given by PSO
Learning rate of BP	–	Given by PSO
Population of PSO	P	10
Convergence coefficient	κ (Eq. (9))	0.5
Damping coefficient	ω (Eq. (9))	0.95
Velocity coefficient	c_1, c_2 (Eq. (9))	1.0
Biases of units	b_i, b_j (Eq. (1))	0.0
Convergence parameter of RBM	α (Eq. (2))	0.05
Convergence parameter of BP	β (Eq. (7))	0.0005
Convergence period of RBM	K (Eq. (3))	50
Iteration times of BP	L	$100 < L < 5000$
Delay of difference time series	d (Eq. (10))	6

[a] The exploration sizes of units were limited from 1 to 20 due to the computational cost.

compare our model with MLP predictors. In this experiment, 980 data in each block of CATS were used to train the models, and instead of previous predicted data, we used the real data of missed area which had been opened to predict the next (one-ahead) data.

The training methods and codes of DBN used Hinton and Salakhutdinov's online materials [10,22]. The number of units of visible layer and hidden layer were decided by the PSO algorithm, as well as the optimization processing shown in Fig. 2.

3.2. *Prediction and result*

Table 1 shows the values of parameters used in the prediction experiments. To compare with the conventional predictors, a 3-layer MLP was used and the numbers of input layer and hidden layer of the MLP were also decided by PSO algorithm optimally. As a powerful evolutionary tool of optimization, PSO has been improved with various versions. Conveniently, we used the original PSO proposed by Kennedy and Eberhart in [16], and the parameters in Eq. (9) (see Table 1) were set to values as same as them discussed in [16].

The difference time series were obtained by 6 time lags delayed which was shown that seasonal trends were extinguished by a software package “Eviews 4.0” (HIS Inc.) for forecasting and statistical analysis.

3.2.1. *The result of Experiment I*

A comparison of the long-term prediction precision of the proposed model (results of Experiment I) and conventional models, which includes the models of IJCNN'04 prediction competition [19,20], typical neural network (MLP) [2], and traditional linear model ARIMA [21] and our previous predictor SOFNN [6–8] using the evaluation function E_1 as shown by Eq. (12) is given in Table 2. The proposed method, i.e., using the 3-layer DBN and PSO optimization of structure and difference time series, ranked no. 2, i.e. showed better precision than conventional MLP [2], SOFNN [6–8], ARIMA [21], and the worst of IJCNN'04 which used a hierarchical Bayesian learning scheme for autoregressive neural networks method.

Table 2
Results of Experiment I (long-term prediction) and comparison to other prediction models (rank of E_1).

Rank	E_1	Model
1	408	Kalman Smoother (The best of IJCNN'04 [19])
2	**1215**	**DBN (proposed model, difference data used)**
3	1216	ARIMA [21] (difference data used)
4	1246	MLP [2] (difference data used)
5	1247	Hierarchical Bayesian Learning (The worst of IJCNN'04 [19])
6	1254	SOFNN [6–8] (difference data used)
7	1317	MLP with PSO (difference data used)
8	1330	ARIMA [21] (original data used)
9	**1622**	**DBN (proposed model, original data used)**
10	2184	MLP [2] (original data used)
11	2655	MLP with PSO (original data used)
12	3366	SOFNN [6–8] (original data used)

Table 3
Results of Experiment I (long-term prediction) and comparison to other prediction models (rank of E_2).

Rank	E_2	Model
1	222	Ensemble Models (The best of IJCNN'04 [19])
2	**979**	**DBN (proposed model, difference data used)**
3	1137	MLP with PSO (difference data used)
4	1179	MLP [2] (difference data used)
5	1185	SOFNN [6–8] (difference data used)
6	1229	Hierarchical Bayesian Learning (The worst of IJCNN'04 [19])
7	1254	ARIMA [21] (difference data used)
8	1349	ARIMA [21] (original data used)
9	**1490**	**DBN (proposed model, original data used)**
10	2337	MLP [2] (original data used)
11	2694	MLP with PSO (original data used)
12	3866	SOFNN [6–8] (original data used)

A comparison of the long-term prediction precision of the proposed model (results of Experiment I) and conventional models using the evaluation function E_2 as shown by Eq. (13) is given in Table 3. The prediction precision of proposed method also ranked No. 2 as same as in Table 2.

Table 4
Results of Experiment II (one-ahead prediction) and comparison to MLP models (rank of E_1).

Rank	E_1	Model
1	**277**	**DBN (proposed model, original data used)**
2	**482**	**DBN (proposed model, difference data used)**
3	540	MLP with PSO (difference data used)
4	2655	MLP with PSO (original data used)

Table 5
Results of Experiment II (one-ahead prediction) and comparison to MLP models (rank of E_2).

Rank	E_2	Model
1	**210**	**DBN (proposed model, original data used)**
2	**341**	**DBN (proposed model, difference data used)**
3	438	MLP with PSO (difference data used)
4	549	MLP with PSO (original data used)

3.2.2. *The result of Experiment II*

A comparison of the short-term (one-ahead) prediction precision of the proposed model (results of Experiment II) and MLP with PSO method using the evaluation function E_1 as shown by Eq. (12) is given in Table 4. The proposed method shows its priority. The result of using original data was better than difference data is considered as that the delay of 6 to yield the difference time series data did not meet the DBN model, but more suitable to MLP model.

Another comparison of the short-term (one-ahead) prediction precision of the proposed model (results of Experiment II) and MLP with PSO models using the evaluation function E_2 as shown by Eq. (13) is given in Table 5. The prediction precision rank is as same as in Table 4.

According to these prediction experiments results, we confirmed that the proposed prediction model, DBN with RBM using pretraining and fine-tuning learning algorithm and PSO structure decision, worked better than conventional models although it did not reach the level of the best of IJCNN'04 competition model. The detail of learning process and analysis of these experiments will be given in next section.

3.3. *Discussion*

As an example of prediction results, Fig. 5 shows those of Block 1, i.e., $t=981–1000$, by different methods in Experiment I (long-term prediction), meanwhile, Fig. 6 shows the results in Experiment II (short-term prediction). The names of curves are defined as follows:

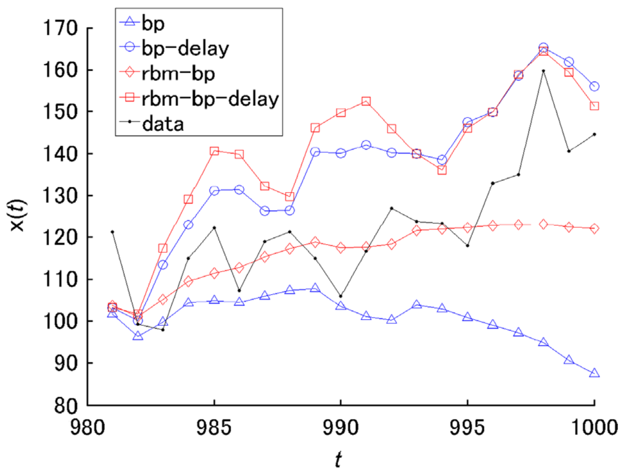

Fig. 5. The long-term prediction results of Block 1 by different methods.

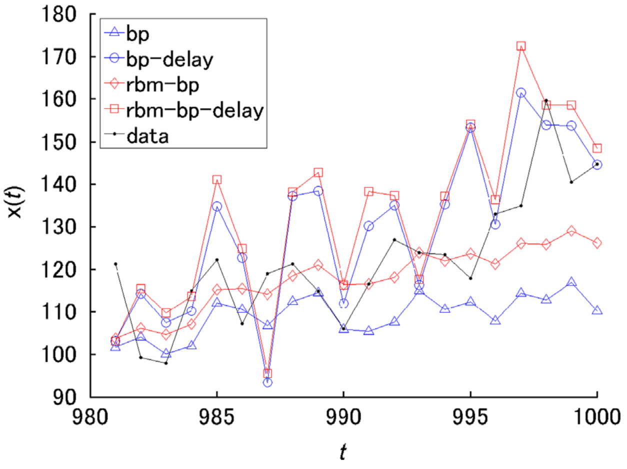

Fig. 6. The short-term prediction results of Block 1 by different methods.

Table 6
Structures and prediction results of DBN and MLP with PSO for the first data block (when data of $t=1, 2, \ldots, 980$ are known, data $t=981$–1000 are predicted).

Structure and Evaluation/Model	bp	bp-delay	**rbm-bp**	**rbm-bp-delay**
The number of units (input–hidden–output)	20–10–1	20–11–1	**20–18–1**	**20–13–1**
Learning rates	0.96	0.46	**0.47, 0.99**	**0.91, 0.96**
Iterations	564	157	**350, 300**	**250, 250**
Learning MSE	129.72	136.55	**135.67**	**131.52**
Long-term prediction MSE	792.49	341.06	**161.92**	**371.39**
One-ahead prediction MSE	334.01	258.70	**134.86**	**337.13**

"bp": MLP with BP learning algorithm and PSO using the original time series ("MLP with PSO (original data used) " in Tables 2–5);
"bp-delay": MLP with BP learning algorithm and PSO using the difference time series ("MLP with PSO (difference data used)" in Tables 2–5);
"rbm-bp": DBN of RBM with RBM learning and BP learning algorithm using the original time series ("DBN (proposed model, original data used)" in Tables 2–5);

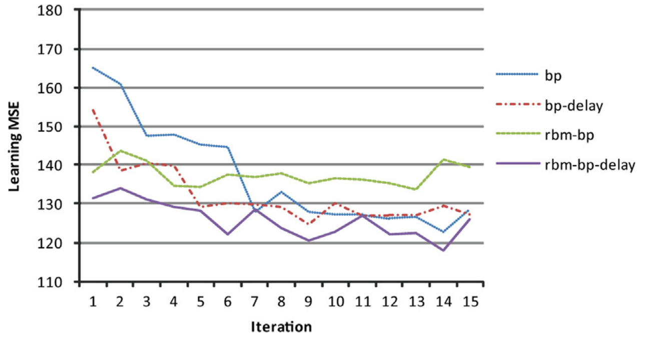

Fig. 7. The change of learning MSE of Block 1 by different methods using PSO algorithm.

"rbm-bp-delay": DBN of RBM with RBM learning and BP learning algorithm using the difference time series ("DBN (proposed model, difference data used)" in Tables 2–5);
"data": The original time series.

Among these forecasting results, "rbm-bp" showed the highest prediction precision.

The detail prediction errors (MSEs) and the structure of DBN obtained by PSO algorithm for the first block data are shown in Table 6. In Table 6, the method with the lowest learning MSE 129.72 is "bp", i.e., the MLP with PSO method using original data, however, the proposed method ("rbm-bp-delay") using difference data shows the highest prediction precision of long-term prediction with 161.92. The best one-ahead prediction of this block 134.86 was also given by the proposed DBN predictor using difference time series. There is a dilemma which can be observed from Table 6. The "Learning MSE" showed the original data yielded lower errors for MLP model ("bp" in Table 6), whereas the difference data had a better results of learning for DBN model ("rbm-bp-delay" in Table 6). However, prediction MSE including long-term prediction and one-ahead prediction for both MLP and DBN models suggested the opposite selections of different time series. "bp-delay" which used the difference data for MLP and "rbm-bp" which used the original data provided higher prediction precisions. So when we decide which kind of data should be used, learning MSE cannot be referred as an index. To solve this problem, cross-validation method may be used, that is, to separate training data to several sub-blocks then evaluating the prediction effectiveness of the different data (difference data or original data). In this paper, we investigated all cases of them for different prediction methods and the final results are listed in Tables 2–5.

The structures of other 4 blocks (input–hidden–output) decided by PSO were:

Block 2: 11–4–1, 19–18–1, 11–3–1, 20–19–1;
Block 3: 18–7–1, 19–13–1, 8–13–1, 19–7–1;
Block 4: 17–11–1, 19–8–1, 18–13–1, 20–7–1;
Block 5: 18–9–1, 20–9–1, 6–7–1, 20–14–1

for MLP (bp), MLP using the difference time series (bp-delay), DBN (rbm-bp), and DBN using the difference time series (rbm-bp-delay), respectively. These different numbers of units corresponding to different training data showed the activity of the PSO architecture optimizations.

The change of Block 1's learning MSE $((1/980)\sum_{t=1}^{980}(x_t-\hat{x}_t)^2)$ during the iteration of PSO algorithm which was used in different models is shown in Fig. 7. Four prediction models, i.e., "bp" (which means conventional MLP model with BP learning algorithm and original data), "bp-delay" (which means conventional MLP model with BP learning algorithm and difference data), "rbm-bp" (the proposed model in this paper and using original data) and "rbm-bp-delay" (using the proposed model and difference data), showed their progress during the optimization process. So iterations of the optimization of structures of ANNs improved the prediction error.

However, just as the overfitting problem exists in the function approximation of ANN models, a good learning (training) result of an approximator may not mean a robust predictor. Though learning MSEs reduced according to the iteration times in Fig. 7, their prediction MSEs were NOT guaranteed. Fig. 8 shows the change of prediction MSE by different methods using PSO algorithm. "bp" and "rbm-bp" present the descent of prediction errors $((1/20)\sum_{t=981}^{1000}(x_t-\hat{x}_t)^2)$ according to the iterations of PSO algorithm, whereas "bp-delay" and "rbm-bp-delay" do not work well during the evolution process though the learning results for approximation showed in Fig. 7 are acceptable. This phenomenon suggests that the finish condition of the structure optimization (Fig. 2 and Step 5 of PSO algorithm) which observes the change of learning MSE needs to be improved, however, we remain this problem in the study of future.

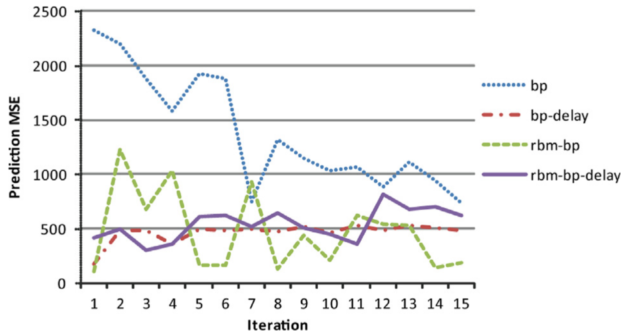

Fig. 8. The change of prediction MSE of Block 1 by different methods using PSO algorithm.

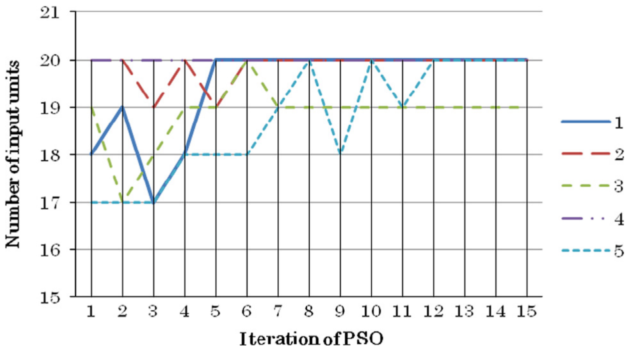

Fig. 9. Changes of the number of units of the input layer in RBM1 (i.e., the input layer of RBM2) for 5 blocks different time series data decided by the PSO algorithm.

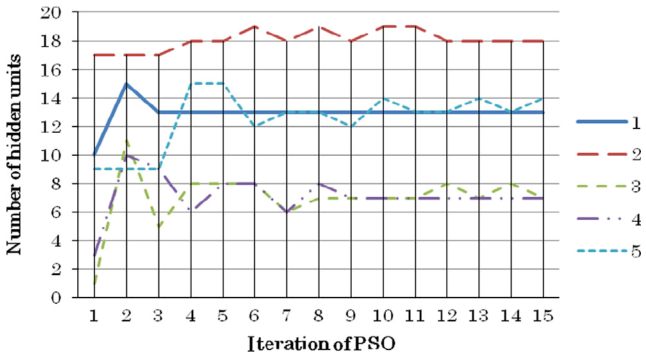

Fig. 10. Changes of the number of units of the hidden layer in RBM1 (i.e., the input layer of RBM2) for 5 blocks different time series data decided by the PSO algorithm.

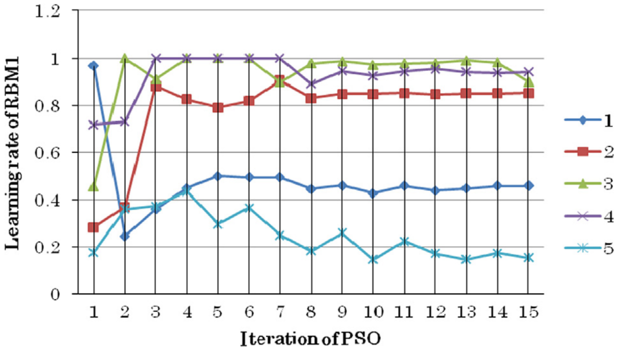

Fig. 11. Changes of the learning rate for RBM1 for 5 blocks different time series data according to the iteration of PSO.

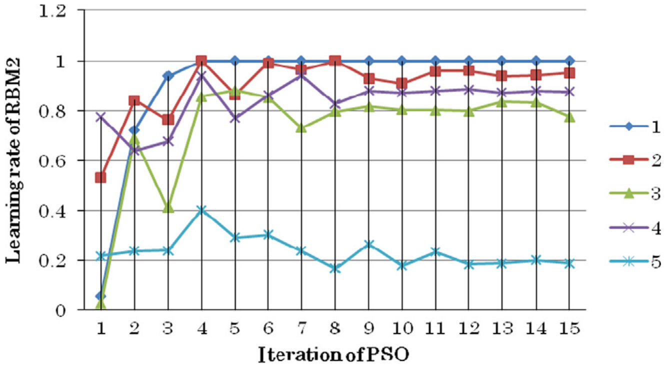

Fig. 12. Changes of the learning rate for RBM2 for 5 blocks different time series data according to the iteration of PSO.

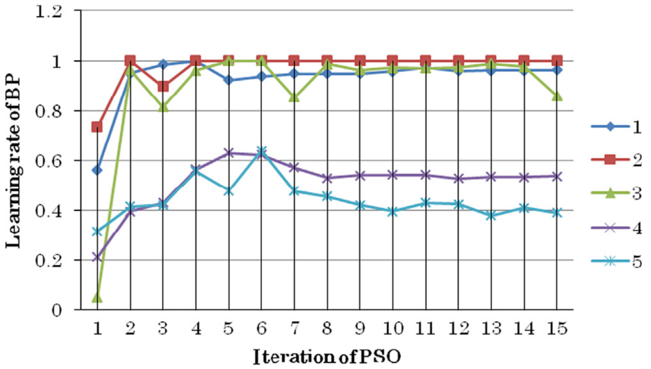

Fig. 13. Changes of the learning rate for DBN during BP's fine-tuning for 5 blocks different time series data according to the iteration of PSO.

The number of units of RMBS and the learning rates decided by PSO when the difference data of 5 blocks used are shown in Figs. 9–13. From Fig. 9, we can see that the input units were fixed to 20 or 19 (Block 3 only), which suggests that more input units used better prediction. So another problem in the future is to investigate the limitation of the number of input units for different time series. Meanwhile, the number of hidden units of RBM1, which were visible units of RBM2, scattered from 7 to 18 at the end of the evolutionary optimization. Considering that the hidden units represent the feature of input vector, these results of Fig. 10 suggested that the adopted structure optimization method worked well because the features of different time series blocks might be different.

In Fig. 11, the change of learning rate of RBM1 for 5 blocks data during optimization is shown. Learning rates for Blocks 2, 3 and

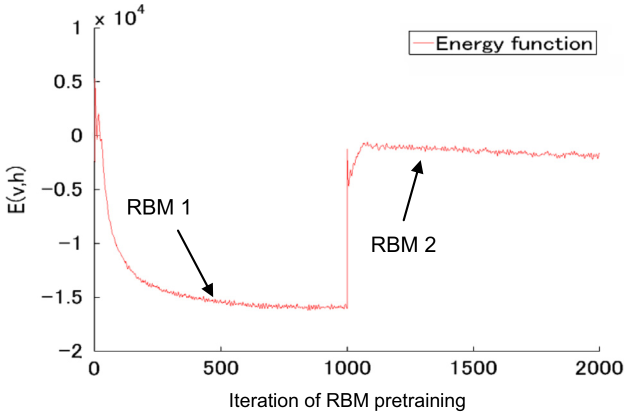

Fig. 14. The change of network energy of RBMs during DBN pretraining.

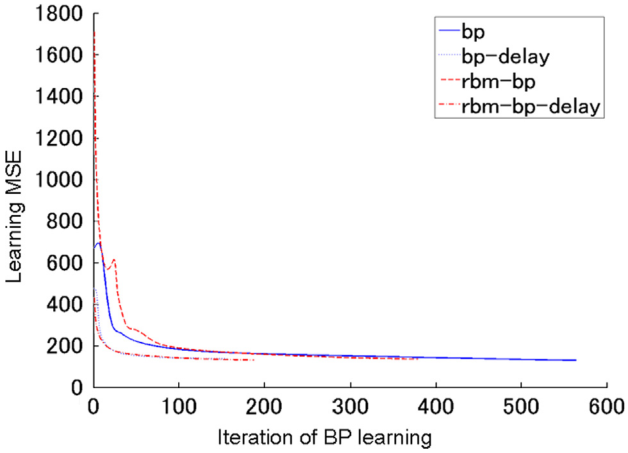

Fig. 15. The change of learning MSE of different models using the BP learning algorithm (fine-tuning of DBN).

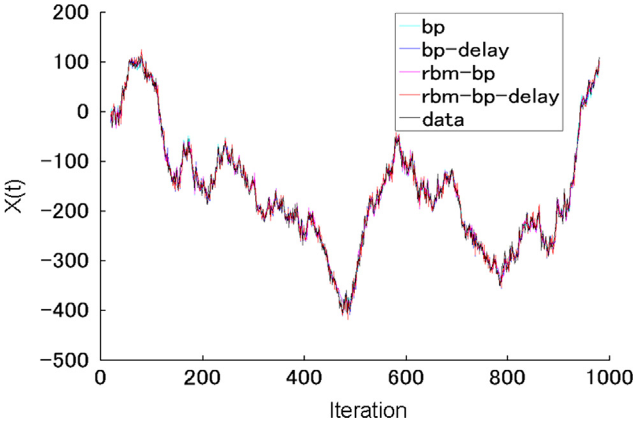

Fig. 16. Approximation results of the different trained prediction models using training samples of Block 1.

4 showed higher values over 0.853 at the end of optimization, meanwhile Blocks 1 and 5 kept lower values which were 0.459 and 0.155 respectively.

In Fig. 12, the change of learning rate of RBM2 for 5 blocks is shown. Except Block 5 kept the lowest value 0.190, learning rates of others were larger than 0.775.

In Fig. 13, the change of learning rate of BP training during the PSO iterations is shown. Blocks 1–3 used higher learning rates which were larger than 0.850, meanwhile Blocks 4 and 5 with 0.547 and 0.391.

Table 7
Learning MSE of different methods for 5 blocks data.

Rank	MSE	Model
1	**125**	**DBN (proposed model, difference data used)**
2	126	MLP with PSO (difference data used)
3	**146**	**DBN (proposed model, original data used)**
4	152	MLP with PSO (original data used)

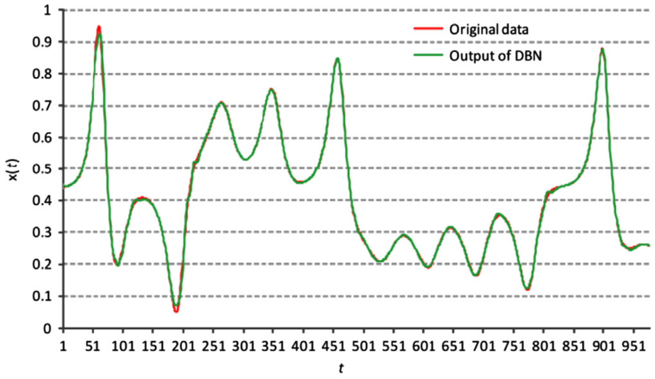

Fig. 17. Learning results of Lorenz chaos using the proposed method.

These different learning rates for different time series and different training methods suggested the necessary of the optimization adoption for the proposed DBN forecasting model.

An example of the pretraining process (in the case of data Block 1) is shown in Fig. 14. The RBMs energy curves reduced according to the RBMs pretraining iterations (Eqs. (1)–(6)). The training of RBM1 stopped at 999 times and from 1000 to 2000 is the change of $E(\mathbf{v}, \mathbf{h})$ of RBM2. We can observe that the network energy of RBM1 showed a sudden descent and converged well. This suggests that the feature of input data was successfully “encoded” to the hidden layer, meanwhile, the situation of RBM2 showed that “the feature of features of input” was not easy to be encoded in the deeper level of DBN.

In Fig. 15, an example of fine-tuning process, i.e., using the BP learning algorithm, observed by the MSE during learning vs. training iteration is shown. All methods converged according to Eq. (7), so they showed different iteration times to finish their training.

Finally, to observe the training results of the different models, the output of them using training data of Block 1 is shown in Fig. 16. In this case, differences between those models results are not significant as one can confirm the learning MSEs are from 129.72 to 236.55 listed in Table 6. The total 5 blocks MSE (given as Eq. (12)) when the learning process converged as Eq. (7) defined are shown in Table 7. Unlike the learning MSE of Block 1 listed in Table 6, the proposed models showed the higher learning performance than MLP models.

3.4. Prediction of chaotic time series

Deterministic chaos has been researched since the end of 19th century, and developed as a new field of complex systems. For the chaotic time series are decided by deterministic formulae, they are appropriate benchmarks to evaluate forecasting methods [6–8,23,24]. Lorenz chaos [23] is well-known with its “butterfly effect” which indicates the sensitive dependence on initial conditions of chaos. Long-term prediction is almost impossible for this reason, however, short-term prediction, especially one-ahead prediction of chaotic time series can be realized if a predictor approximates the nonlinear system enough.

Table 8
Learning RMSE of different methods for Lorenz chaos (× 10^{-3}).

Rank	*RMSE*	Model
1	**5.57**	**DBN (proposed model)**
2	23.00	MLP [2]
3	27.40	SOFNN [6–8]
4	42.66	HYBRID (ARIMA+SOFNN)
5	54.40	HYBRID (ARIMA+MLP) [17]
6	112.67	ARIMA [21]

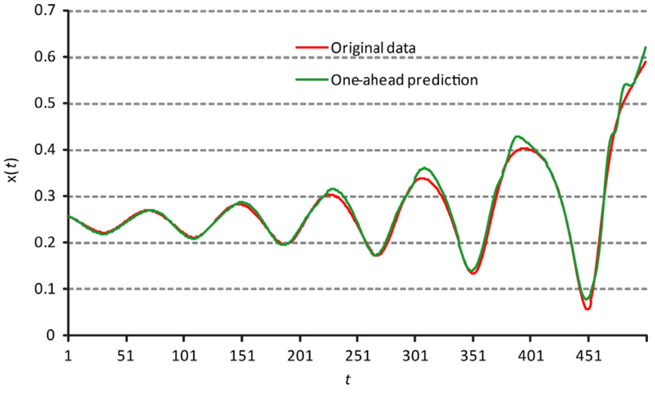

Fig. 18. Short-term (one-ahead) prediction results of Lorenz chaos using the proposed method.

Table 9
One-ahead prediction RMSE of different methods for Lorenz chaos (× 10^{-3}).

Rank	*RMSE*	Model
1	**10.20**	**DBN (proposed model)**
2	24.84	MLP [2]
3	26.42	SOFNN [6–8]
4	31.37	HYBRID (ARIMA+MLP) [17]
5	46.30	HYBRID (ARIMA+SOFNN)
6	160.85	ARIMA [21]

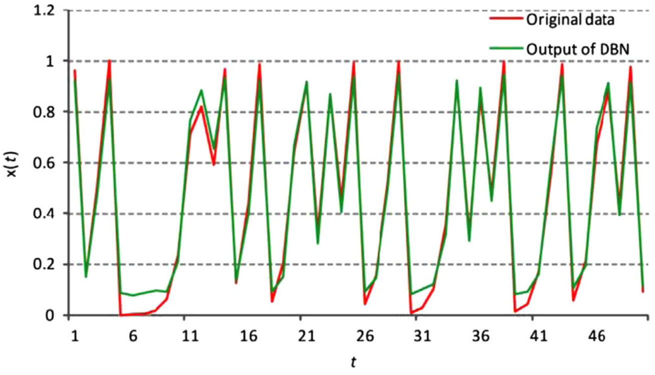

Fig. 19. Learning results of logistic map using the proposed method (The first 50 step of 980 sample data).

We used a time series of Lorenz chaos, as shown in Fig. 17 which had 1000 data with values from 0.0 to 1.0 ("Original data" in Fig. 17), to train the proposed DBN. The parameters of the model were set as same as in Table 1 except that the exploration range of PSO for determining the number of input units and hidden units was enlarged to be from 1 to 100. The optimal structure of DBN was obtained as "26–80–1", i.e., 26 input units, 80 hidden units and 1 output unit. The learning rates of RBM1, RBM2, and BP were 0.39, 0.07, and 0.51, respectively.

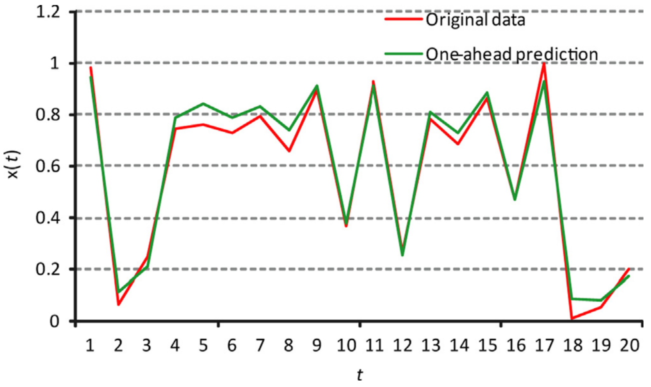

Fig. 20. Short-term (one-ahead) prediction results of logistic map using the proposed method (test data from 980 to 1000).

Learning result, i.e., approximation given by DBN is shown in Fig. 17, and the precision evaluation was given by root mean squared errors (RMSE) shown in Table 8. Comparing with other ANN models such as MLP [2], SOFNN [6–8], and classical linear model ARIMA, and hybrid models such as ARIMA with SOFNNN and ARIMA with MLP [17], the proposed DBN had the lowest approximation error.

After training of DBN, we used it to perform one-ahead prediction for the future 500 data of the Lorenz time series. The prediction result is shown in Fig. 18 and the prediction precision was evaluated with RMSE as shown in Table 9. The proposed method also stands on the top position comparing with the other forecasting methods.

We also used other chaos such as logistic map to examine the effectiveness of our approximation and prediction model. The structure of DBN obtained by PSO was "8–14–1" and the learning rates were 0.87, 0.68, and 0.50 for RBM1, RBM2, and BP respectively. Fig. 19 shows a part of learning results of 980 data of logistic map, meanwhile 20 future data were predicted (one-ahead prediction) as shown in Fig. 20. Both approximation and prediction worked but higher precision needs to be obtained by improving the proposed method in the future.

4. Conclusions

A novel neural network model for time series forecasting was proposed in this paper. The proposed model is a kind of deep believe network (DBN) composed by two restricted Boltzmann machines (RBMs). The structure of RBMs was optimized by a classical particle swarm optimization (PSO) algorithm. Preprocessing to the original time series data were also performed to obtain difference time series data which were smoothed data and more appropriated for neural network prediction models.

Using CATS benchmark, the proposed model was confirmed its priority to conventional neural network models such as multilayer perceptron (MLP), self-organizing fuzzy neural network (SOFNN), and the mathematical linear model ARIMA.

The approximation and prediction abilities of DBN were also confirmed by using different chaotic time series such as Lorenz chaos and logistic map, and it also works better than other well-known conventional methods.

The prediction precision of CATS benchmark was not higher than the best record of IJCNN'04 prediction competition participant methods (where one winner used a Kalman Smoother Model and another winner proposed Ensemble Models). However, as a new original predictor, the proposed method can be expected to raise its prediction precision in the future.

Acknowledgment

We would like to thank dear Reviewers and Editors for their appropriate advices during the revision of this paper, and we also thank Dr. Shingo Mabu and Mr. Shintaro Tokuda for their warmly support to this research.

References

[1] S. Crone, K. Nikolopoulos, Results of the NN3 neural network forecasting competition, in: Proceedings of the 27th International Symposium on Forecasting Program, 2007, pp. 129.

[2] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representation by backpropagating errors, Nature 232 (1986) 533–536.

[3] R. Hecht-Nielsen, Theory of the backpropagation neural network, in: Proceedings of the International Joint Conference on Neural Networks, vol. 1, 1989, pp. 593–611.

[4] V. Pacelli, V. Bevilacqua, M. Azzollini, An artificial neural network model to forecast exchange rates, J. Intell. Learn. Syst. Appl.3 (2011) 57–69.

[5] G. Panchal, A. Ganatra, Y.P. Kosta, D. Panchal, Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers, Int. J. Comput. Theory Eng. 3 (2) (2011) 332–337.

[6] T. Kuremoto, M. Obayashi, A. Yamamoto, K. Kobayashi, Predicting chaotic time series by reinforcement learning, in: Proceedings of the 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS'03), 2003.

[7] T. Kuremoto, M. Obayashi, K. Kobayashi, Forecasting time series by SOFNN with reinforcement learning, in: Proceedings of the 27th International Symposium on Forecasting (ISF 2007) Program, 2007, pp. 99.

[8] T. Kuremoto, M. Obayashi, K. Kobayashi, Neural forecasting systems, in: C. Weber, M. Elshaw, N.M. Mayer (Eds.), Reinforcement Learning: Theory and Applications, INTECH, Rijeca, Croatia, 2008, pp. 1–20.

[9] G.E. Hinton, S. Osindero, Y.W. The, A faster learning algorithm for deep belief nets, Neural Comput. 1 (7) (2006) 1527–1544.

[10] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (2006) 504–507.

[11] Y. Bengio, P. Lamlin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Proceedings of Advances in Neural Information Processing Systems (NIPS 2006), vol. 19, 2007, pp. 1–8.

[12] K. Chen, A. Salman, Learning speaker-specific characteristics with a deep neural architecture, IEEE Trans. Neural Netw. 22 (11) (2011).

[13] N.L. Roux, Y. Bengio, Representational power of restricted Boltzmann machines and deep belief networks, Neural Comput. 20 (6) (2008) 1631–1649.

[14] D.H. Ackley, G.E. Hinton, T.J. Sejnowski, A learning algorithm for Boltzmann machines, Cogn. Sci. 9 (1985) 147–169.

[15] G.E. Hinton, T.J. Sejnowski, Learning and relearning in Boltzmann machines, in: D.E. Rumelhart, J.L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations, MIT Press, Cambridge, MA, 1986.

[16] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.

[17] G.P. Zhang, Time series forecasting using a hybrid ARIMA and neural network model, Neurocomputing 50 (2003) 159–175.

[18] E. Gardner, E. McKenzie, Seasonal exponential smoothing with damped trends, Manag. Sci. 35 (3) (1989) 372–376.

[19] A. Lendasse, E. Oja, O. Simula, M. Verleysen, Time series prediction competition: the CATS benchmark, in: Proceedings of the International Joint Conference on Neural Networks, 2004, pp. 1615–1620.

[20] A. Lendasse, E. Oja, O. Simula, M. Verleysen, Time series prediction competition: the CATS benchmark, Neurocomputing 70 (2007) 2325–2329.

[21] G.E.P. Box, G. Jenkins, Time Series Analysis, Forecasting and Control, Cambridge University Press, Cambridge, 1976.

[22] G.E. Hinton, R.R. Salakhutdinov, Supporting online material, 2006. ⟨www.sciencemag.org/cgi/content/313/5786/504/DC1⟩ Matlab Code.

[23] E.N. Lorenz, Deterministic nonperiodic flow, J. Atmos. Sci. 20 (1963) 130–141.

[24] T. Kuremoto, M. Obayashi, K. Kobayashi, Nonlinear prediction by reinforcement learning, in: Proceedings of the 1st International Conference on Intelligent Computing (ICIC 2005), D.S. Huang, X.P. Zhang, G.B. Huang, (Eds.), LNCS, Springer, vol. 3644, 2005, pp.1085–1094.

Takashi Kuremoto received the B.E. degree in System Engineering at University of Shanghai for Science and Technology, China in 1986, and M.E. degree at Yamaguchi University, Japan in 1996. He worked as a System Engineer at Research Institute of Automatic Machine of Beijing from 1986 to 1992 and is currently an Assistant Professor in Division of Computer Science & Design Engineering, Graduate School of Science and Engineering at Yamaguchi University, Japan. He was an Academic Visitor of School of Computer Science, The University of Manchester, UK in 2008. His research interests include bioinformatics, machine learning, complex systems, forecasting and swarm intelligence. He is a member of IEICE, IEEE, IIF and SICE.

Shinsuke Kimura received the B.E. degree in Department of Information Science in 2010, and Engineering and M.E. degree in Graduate School of Science and Engineering in 2012, in Yamaguchi University, Japan. He is currently with Tokyo Electron Kyushu Ltd.

Kunikazu Kobayashi received B.E., M.E., and Ph.D. degrees in Electronic Engineering, Computer Science, and Computer Engineering from Yamaguchi University, Japan, respectively. From 1994 to 2012, he was an Assistant Professor at Division of Computer Science & Design Engineering, Yamaguchi University. He was a Visiting Research Professor at Department of Biomedical Engineering, University of Houston, USA in 2010. He was awarded the Highest Honor Student Award from Yamaguchi University and the Best Paper Award at ANNIE in 1994. Currently, he is an Associate Professor at School of Information Science and Technology, Aichi Prefectural University, Japan. His research interests include neural networks (especially wavelet neural networks), Bayesian method, reinforcement learning, multi-agent system, and biomedical data analysis. He is a member of IEEE, IEEJ, SICE, and IEICE and also a member of research committee on real application oriented machine learning of IEEJ.

Masanao Obayashi received the B.S. degree in Electrical Engineering from Kyushu Institute of Technology, Japan, in 1975, the M.S. and the Ph.D. degrees in Engineering from Kyushu University, Japan, in 1977 and 1997, respectively. From 1977 to 1989, he was with Hitachi Ltd. From 1989 to 1998, he was with Kyushu University, where he was a Research Associate and an Associate Professor. From 1998 he is with Yamaguchi University, Japan, since July 2001 he has been a Professor in the Faculty of Engineering, Yamaguchi University. He is now with the Graduate school of Science and Engineering of the same university. His current research interests are intelligent computing and intelligent control. Prof. Obayashi is a member of SICE, IEEJ, JSAI, JSFTII and IEEE.