We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Chapter

Training Deep Neural Networks with Reinforcement Learning for Time Series Forecasting

Takashi Kuremoto, Takaomi Hirata, Masanao Obayashi, Shingo Mabu and Kunikazu Kobayashi

Abstract

As a kind of efficient nonlinear function approximators, artificial neural networks (ANN) have been popularly applied to time series forecasting. The training method of ANN usually utilizes error back-propagation (BP) which is a supervised learning algorithm proposed by Rumelhart et al. in 1986; meanwhile, authors proposed to improve the robustness of the ANN for unknown time series prediction using a reinforcement learning algorithm named stochastic gradient ascent (SGA) originally proposed by Kimura and Kobayashi for control problems in 1998. We also successfully use a deep belief net (DBN) stacked by multiple restricted Boltzmann machines (RBMs) to realized time series forecasting in 2012. In this chapter, a state-of-the-art time series forecasting system that combines RBMs and multilayer perceptron (MLP) and uses SGA training algorithm is introduced. Experiment results showed the high prediction precision of the novel system not only for benchmark data but also for real phenomenon time series data.

Keywords: artificial neural networks (ANN), deep learning (DL), reinforcement learning (RL), deep belief net (DBN), restricted Boltzmann machine (RBM), multilayer perceptron (MLP), stochastic gradient ascent (SGA)

1. Introduction

Artificial neural networks (ANN), which are mathematical models for function approximation, classification, pattern recognition, nonlinear control, etc., have been successfully applied in the field of time series analysis and forecasting instead of linear models such as 1970s ARIMA [1] since 1980s [2–7]. In [2], Casdagli used a radial basis function network (RBFN) which is a kind of feed-forward neural network with Gaussian hidden units to predict chaotic time series data, such as the Mackey-Glass, the Ikeda map, and the Lorenz chaos in 1989. In [3, 4], Lendasse et al. organized a time series forecasting competition for neural network prediction methods with a five-block artificial time series data named CATS since 2004. The goal of CATS competition was to predict 100 missing values of the time series data in five sets which included 980 known values and 20 successive unknown values in each set (details are in Section 3.1). There were 24 submissions to the competition, and five kinds of methods were selected by the IJCNN2004: filtering techniques including Bayesian methods, Kalman filters, and so on; recurrent neural networks

IntechOpen

(RNNs); vector quantization; fuzzy logic; and ensemble methods. As the comment of the organizers, the different prediction precisions were reported though the similar prediction methods were used for the know-how and experience of the authors. So the development of time series forecasting by ANN is still on the way.

As a kind of classifiers or a kind of function approximators, the advances of the ANN are bought out by the nonlinear transforms to the input space. In fact, units (or neurons) with nonlinear firing functions connected to each other usually produce higher dimensional output space and various feature spaces in the networks. Additionally, as a connective system, it is not necessary to design fixed mathematical models for different nonlinear phenomena, but adjusting the weights of connections between units. So according to the report of NN3—Artificial Neural Networks and Computational Intelligence Forecasting Competition [5], there have been more than 5000 publications of time series forecasting using ANN till 2007.

To find the suitable parameters of ANN, such as weights of connections between neurons, error back-propagation (BP) algorithm [6] is generally utilized in the training process of ANN. However, due to every sample data (a pair of the input data and the output data) is used in the BP method, noise data influences the optimization of the model, and robustness of the model becomes weak for unknown input. Another problem of ANN models is how to determine the structure of the network, i.e., the number of layers and the number of neurons in each layer. To overcome these problems of BP, Kuremoto et al. [7] adopted a reinforcement learning (RL) method "stochastic gradient ascent (SGA)" [8] to adjust the connection weights of units and the particle swarm optimization (PSO) to find the optimal structure of ANN. SGA, which is proposed by Kimura and Kobayshi, improved Williams' REINFORCE [9], which uses rewards to modify the stochastic policies (likelihood). In SGA learning algorithm, the accumulated modification of policies named "eligibility trace" is used to adjust the parameters of model (see Section 2). In the case of time series forecasting, the reward of RL system can be defined as a suitable error zone to instead of the distance (error) between the output of the model and the teach data which is used in BP learning algorithm. So the sensitivity to noise data is possible to be reduced, and the robustness to the unknown data may be raised. As a deep learning method for time series forecasting, Kuremoto et al. [10] firstly applied Hinton and Salakhutdinov's deep belief net (DBN) which is a kind of stacked auto-encoder (SAE) composed by multiple restricted Boltzmann machines (RBMs) [11]. An improved DBN for time series forecasting is proposed in [12], which DBN is composed by multiple RBMs and a multilayer perceptron (MLP) [6]. The improved DBN with RBMs and MLP [6] gives its priority to the conventional DBN [5] for time series forecasting due to the continuous output unit is used; meanwhile the conventional one had a binary value unit in the output layer.

As same as the RL method, SGA adopted to MLP, RBFN, and self-organized fuzzy neural network (SOFNN) [7]; the prediction precision of DBN utilized SGA may also be raised comparing to the BP learning algorithm. Furthermore, it is available to raise the prediction precision by a hybrid model which forecasts the future data by the linear model ARIMA at first and modifying the forecasting by the predicted error given by an ANN which is trained by error time series [13, 14].

In this chapter, we concentrate to introduce the DBN which is composed by multiple RBMs and MLP and show the higher efficiency of the RL learning method SGA for the DBN [15, 16] comparing to the conventional learning method BP using the results of time series forecasting experiments. Kinds of benchmark data including artificial time series data CATS [3], natural phenomenon time series data provided by Aalto University [18], and TSDL [18] were used in the experiments.

2. The DBN model for time series forecasting

2.1 The structure of the model

The model of time series forecasting is given as the following:

$$x_{t+1} = f(x_t, x_{t-1}, \ldots, x_{t-n+1}) \tag{1}$$

Denote $t = 1, 2, 3, \ldots$, where T is the time, n is the dimensionality of the input of function $f(x)$, x_t is the time series data, and x_{t+1} is unknown data in the future as well as the output of model.

A deep belief net (DBN) composed by restricted Boltzmann machines (RBMs) and multilayer perceptron (MLP) is shown in **Figure 1**.

2.2 RBM

Restricted Boltzmann machine (RBM) is a kind of probabilistic generative neural network which composed by two layers of units: visible layer and hidden layer (see **Figure 2**).

Units of different layers connect to each other with weights $w_{ij} = w_{ji}$, where $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$ are the numbers of units of visible layer and hidden layer, respectively. The outputs of units v_i, h_j are binary, i.e., 0 or 1, except for the initial value of visible units which is given by the input data. The probabilities of 1 of a visible unit and a hidden unit are according to the following:

$$p(h_j = 1|v) = \frac{1}{1 + \exp\left(-b_j - \sum_{i=1}^{n} w_{ji} v_i\right)} \tag{2}$$

$$p(v_i = 1|\mathrm{h}) = \frac{1}{1 + \exp\left(-b_i - \sum_{j=1}^{m} w_{ij} h_j\right)} \tag{3}$$

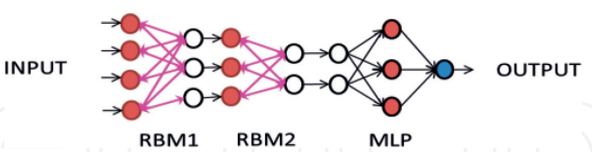


Figure 1.
The structure of DBN for time series forecasting.

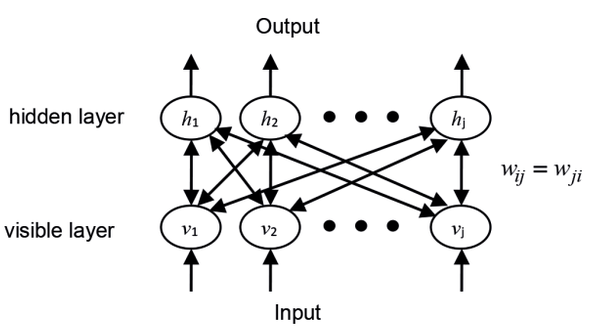


Figure 2.
The structure of RBM.

Here b_i, b_j are the biases of units. The learning rules of RBM are given as follows:

$$\Delta w_{ij} = \varepsilon\left(< v_i h_j >_{\mathbf{data}} - < v_i h_j >_{\mathbf{model}} \right) \tag{4}$$

$$\Delta b_i = \varepsilon\left(< v_i > - < \tilde{v}_i > \right) \tag{5}$$

$$\Delta b_j = \varepsilon\left(< h_j > - < \tilde{h}_j > \right) \tag{6}$$

where $0 < \varepsilon < 1$ is a learning rate, $p_{ij} = < v_i h_j >_{\mathbf{data}}$, $p'_{ij} < v_i h_j >_{\mathbf{model}}$ and $< v_i >$, $< h_j >$ indicate the expectations of the first Gibbs sampling (k = 0), and $< \tilde{v}_i >$, $< \tilde{h}_j >$ the kth Gibbs sampling, and it works when k = 1.

2.3 MLP

Multilayer perceptron (MLP) is the most popular neural network which is generally composed by three layers of units: input layer, hidden layer, and output layer (see **Figure 3**).

The output of the unit $y = f(z)$ and unit $z_k = f(\mathrm{x})$ is given as the following logistic sigmoid functions:

$$f(y) = \frac{1}{1 + \exp\left(-\sum_{j=1}^{K+1} w_j z_j\right)} \tag{7}$$

$$f(z_j) = \frac{1}{1 + \exp\left(-\sum_{i=1}^{n+1} v_{ji} x_i\right)} \tag{8}$$

Here n is the dimensionality of the input, K is the number of hidden units, and $x_{n+1} = 1.0$, $z_{K+1} = 1.0$ are the support units of biases $v_{j(n+1)}$, w_{K+1}.

The learning rules of MLP using error back-propagation (BP) method [5] are given as follows:

$$\Delta w_j = -\varepsilon\left(y - \tilde{y}\right) y(1-y) z_j \tag{9}$$

$$\Delta v_{ji} = -\varepsilon\left(y - \tilde{y}\right) y(1-y) w_j z_j \left(1 - z_j\right) x_i \tag{10}$$

where $0 < \varepsilon < 1$ is a learning rate and $\tilde{y}$ is the teacher signal.

The learning algorithm of MLP using BP is as follows:

Step 1. Observe an input $\mathrm{x}_t = (x_t, x_{t-1}, \ldots, x_{t-n+1})$;

Step 2. Predict a future data $y_t = x_{t+1}$ according to Eqs. (7) and (8).

Step 3. Calculate the modification of connection weights, Δw_j, Δv_{ji} according to Eqs. (9) and (10).

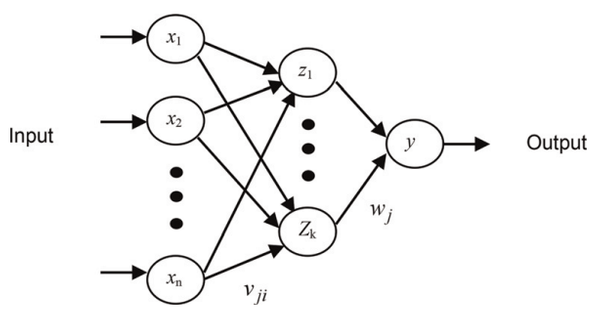


Figure 3.
The structure of MLP.

Step 4. Modify the connections,

$$w_j \leftarrow w_j + \Delta w_j; v_{ji} \leftarrow v_{jj} + \Delta v_{ji};$$

Step 5. For the next time step $t+1$, return to step 1.

2.4 The training method of DBN

As same as the training process proposed in [10], the training process of DBN is performed by two steps. The first one, pretraining, utilizes the learning rules of RBM, i.e., Eqs. (4–6), for each RBM independently. The second step is a fine-tuning process using the pretrained parameters of RBMs and BP algorithm. These processes are shown in **Figure 4** and Eqs. (11)–(13).

$$\Delta w_{ji}^{L} = -\varepsilon\left(\sum_i \frac{\partial E}{\partial w_{ji}^{L+1}} w_{ji}^{L+1}\right)\left(1-h_j^L\right)v_i^L \tag{11}$$

$$\Delta b_j^{L} = -\varepsilon\left(\sum_i \frac{\partial E}{\partial w_{ji}^{L+1}} w_{ji}^{L+1}\right)\left(1-h_j^L\right) \tag{12}$$

$$E = \frac{1}{2}\sum_{t=1}^{T}\left(y_t - \tilde{y}_t\right) \tag{13}$$

In the case of reinforcement learning (RL), the output is decided by a probability distribution, e.g., the Gaussian distribution $y \sim \pi(\mu, \sigma^2)$. So the output units are the mean μ and the variance σ instead of one unit y.

$$\mu = \sum_j w_{\mu j} z_j \tag{14}$$

$$\sigma = \frac{1}{1+\exp\left(-\sum_j w_{\sigma j} z_j\right)} \tag{15}$$

$$\pi\left(\mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right) \tag{16}$$

The learning algorithm of stochastic gradient ascent (SGA) [7] is as follows.

Step 1. Observe an input $\mathrm{x}_t = (x_t, x_{t-1}, ..., x_{t-n+1})$.

Step 2. Predict a future data $y_t = x_{t+1}$ according to a probability $y_t \sim \pi(\mathrm{x}_t, \mathrm{w})$ with ANN models which are constructed by parameters $\mathrm{w}\left(w_{\mu j}, w_{\sigma j}, w_{ij}, v_{ji}\right)$.

Step 3. Receive a scalar reward/punishment r_t by calculating the prediction error:

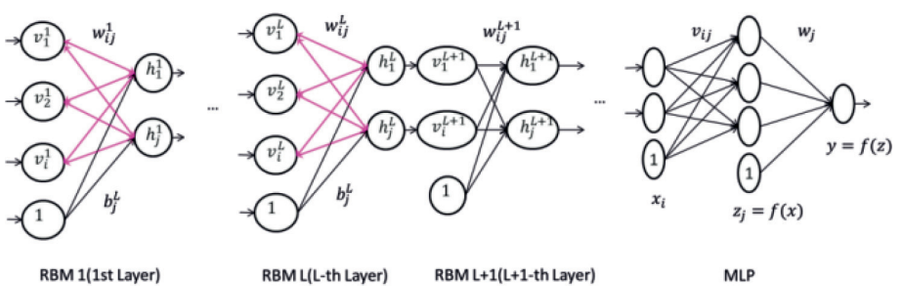


Figure 4.
The training of DBN by BP method.

$$r_t = \begin{cases} 1 & if\ (y_t - \tilde{y}_t)^2 \le \zeta \\ -1 & else \end{cases} \tag{17}$$

where ζ is an evaluation constant greater than or equal to zero.

Step 4. Calculate characteristic eligibility $e_i(t)$ and eligibility trace $\overline{D}_i(t)$:

$$e_i(t) = \frac{\partial}{\partial w_i} \ln \{\pi(\mathrm{x}_t, \mathrm{w})\} \tag{18}$$

$$\overline{D}_i(t) = e_i(t) + \gamma \overline{D}_i(t-1) \tag{19}$$

where $0 \le \gamma < 1$ is a discount factor and w_i denotes ith internal variable of DBN.

Step 5. Calculate the modification $\Delta w_i(t)$:

$$\Delta w_i(t) = (r_t - b)\overline{D}_i(t) \tag{20}$$

where $b \ge 0$ denotes the reinforcement baseline (it can be set as zero).

Step 6. Improve the policy Eq. (16) by renewing its internal variable w_i by Eq. (21):

$$w_i \leftarrow w_i + \varepsilon \Delta w_i \tag{21}$$

where $0 \le \varepsilon \le 1$ is a learning rate.

Step 7. For the next time step $t+1$, return to step 1.

Characteristic eligibility $e_i(t)$, shown in Eq. (18), means that the change of the policy function concerns with the change of system internal variable vector. In fact, the algorithm combines reward/punishment to modify the stochastic policy with its internal variable renewing by Step 4 and Step 5.

The calculation of $e_{w_{\mu ij}}(t)$, $e_{w_{\sigma j}}(t)$, $e_{v_{ij}}(t)$ in MLP part of DBN is induced as follows;

$$e_{w_{\mu ij}}(t) = \frac{y_t - \mu_t}{\sigma_t^2} z_j(t) \tag{22}$$

$$e_{w_{\sigma j}}(t) = \frac{(y_t - \mu_t)^2 - \sigma^2}{\sigma_t^2} (1 - \sigma_t) z_j(t) \tag{23}$$

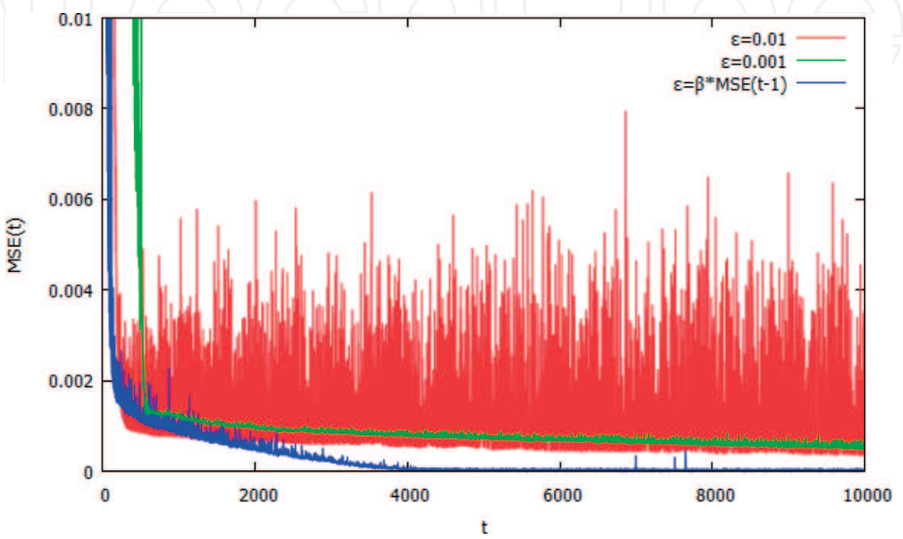


Figure 5.
The learning errors given by different learning rates.

$$e_{v_{ij}}(t) = \left(e_{w_{\mu j}}(t)w_{\mu j} + e_{w_{\sigma j}}(t)w_{\sigma j}\right)\left(1 - z_j(t)\right)x_i(t) \tag{24}$$

The $e_i(t)$ of the RBM of Lth layer in the case of the DBN is given as follows:

$$e^L_{w_{ij}}(t) = \left(\sum_j e^{L+1}_{w_{ij}}(t)w^{L+1}_{w_{ij}}\right)\left(1 - h^L_j\right)v^L_i \tag{25}$$

$$e^L_{b_j}(t) = \left(\sum_k e^{L+1}_{w_{ij}}(t)w^{L+1}_{w_{ij}}\right)\left(1 - h^L_j\right) \tag{26}$$

The learning rate ε in Eq. (21) affects the learning performance of fine-tuning of DBN. Different values to result different training error (mean squared error (MSE)) as shown in **Figure 5**. An adaptive learning rate as a linear function of learning error is proposed as in Eq. (27):

$$\varepsilon = \beta \mathrm{MSE}(t-1) \tag{27}$$

where is $0 \leq \beta$ a constant.

2.5 Optimization of meta-parameters

The number of RBM that constitute the DBN and the number of neurons of each layer affects prediction performance seriously. In [9], particle swarm optimization (PSO) method is used to decide the structure of DBN, and in [13] it is suggested that random search method [16] is more efficient. In the experiment of time series forecasting by DBN and SGA shown in this chapter, these meta-parameters were decided by the random search, and the exploration limits are shown as the following.

- The number of RBMs: [0–3]
- The number of units in each layer of DBN: [2–20]
- Learning rate of each RBM in Eqs. (4)–(6): [10^{-5}–10^{-1}]
- Fixed learning rate of SGA in Eq. (21): [10^{-5}–10^{-1}]
- Discount factor in Eq. (19): [10^{-5}–10^{-1}]
- Coefficient in Eq. (27) [0.5–2.0]

The optimization algorithm of these meta-parameters by the random search method is as follows**:**

Step 1. Set random values of meta-parameters beyond the exploration limitations.

Step 2. Predict a future data $y_t \approx x_{t+1}$ by MLP or DBN using the current weighted connections.

Step 3. If the error between y_t, x_{t+1} is reduced enough, store the values of meta-parameters,
or else if the error is not changed,
stop the exploration,
else return to step 1.

3. The experiments and results

3.1 CATS benchmark time series data

CATS time series data is the artificial benchmark data for forecasting competition with ANN methods [3, 4].This artificial time series is given with 5000 data, among which 100 are missed (hidden by competition the organizers). The missed data exist in five blocks:

- Elements 981 to 1000
- Elements 1981 to 2000
- Elements 2981 to 3000
- Elements 3981 to 4000
- Elements 4981 to 5000

The mean square error E_1 is used as the prediction precision in the competition, and it is computed by the 100 missing data and their predicted values as the following:

$$E_1 = \left\{ \sum_{t=981}^{1000} (y_t - \bar{y}_t)^2 + \sum_{t=1981}^{2000} (y_t - \bar{y}_t)^2 + \sum_{t=2981}^{3000} (y_t - \bar{y}_t)^2 + \sum_{t=3981}^{4000} (y_t - \bar{y}_t)^2 + \sum_{t=4981}^{5000} (y_t - \bar{y}_t)^2 \right\} / 100 \tag{28}$$

where $\bar{y}_t$ is the long-term prediction result of the missed data. The CATS time series data is shown in **Figure 6**.

The prediction results of different blocks of CATS data are shown in **Figure 7**. Comparing to the conventional learning method of DBN, i.e., using Hinton's RBM unsupervised learning method [6, 8] and back-propagation (BP), the proposed method which used the reinforcement learning method SGA instead of BP showed its superiority in the sense of the average prediction precision E_1 (see **Figure 7f**).

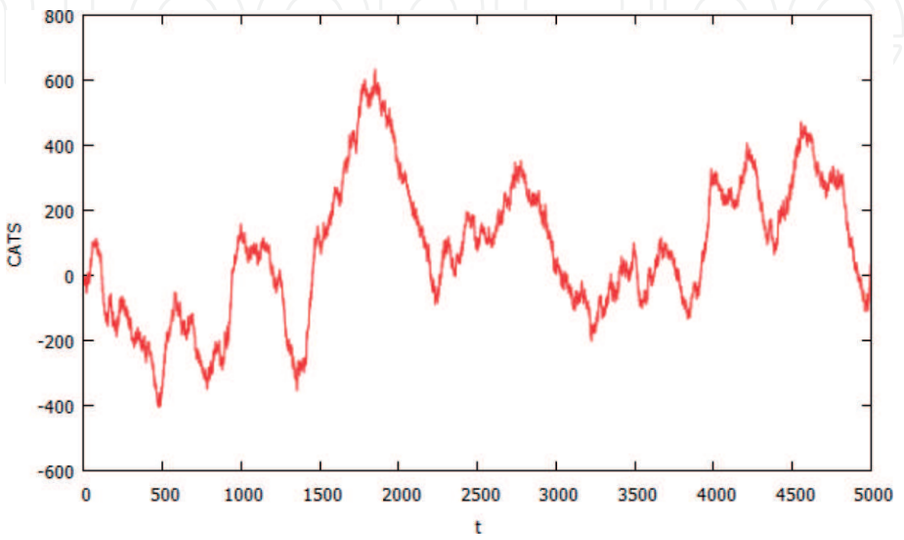


Figure 6.
CATS benchmark data.

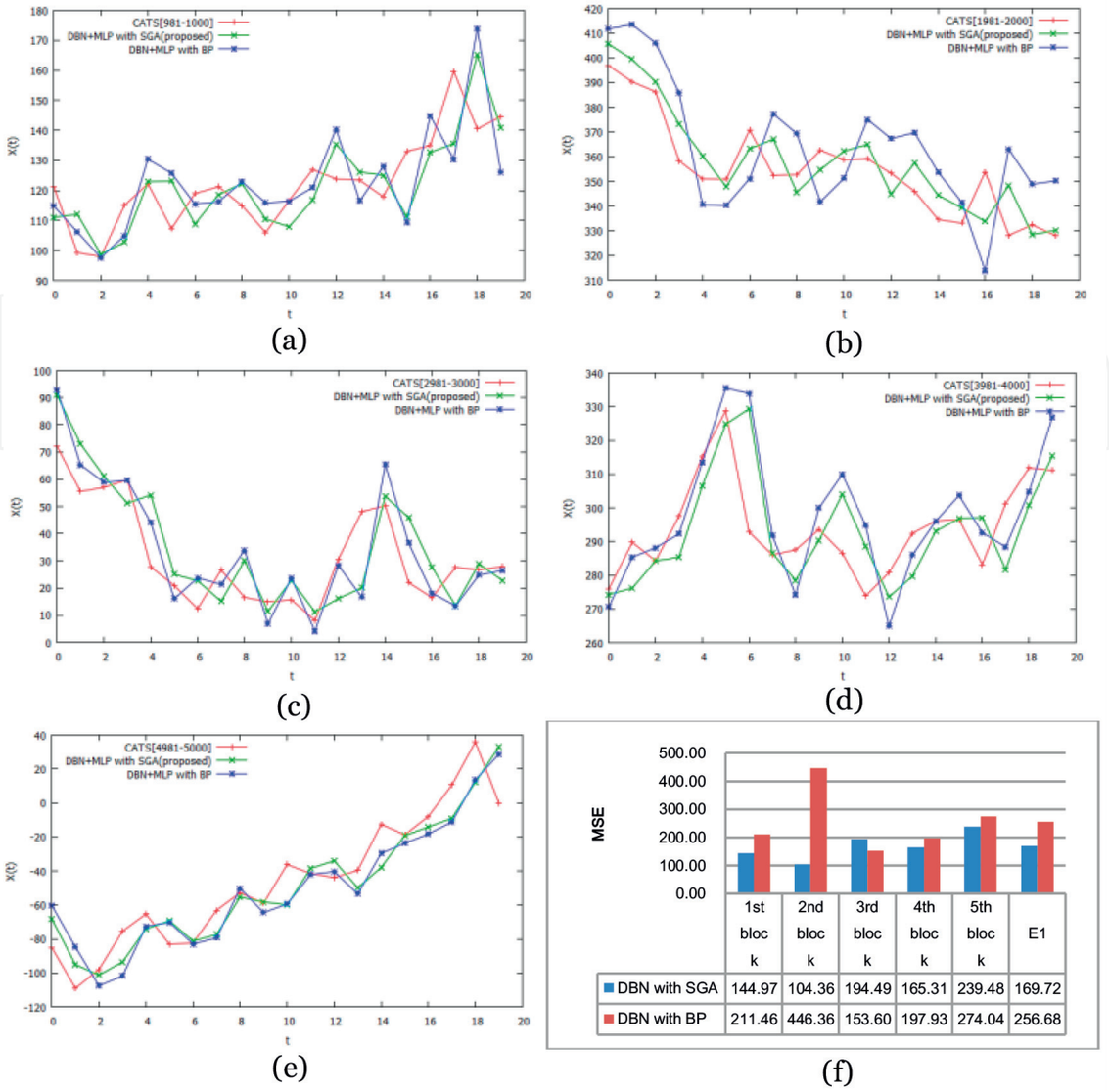


Figure 7.
The prediction results of different methods for CATS data: (a) block 1; (b) block 2; (c) block 3; (d) block 4; (e) block 5; and (f) results of the long-term forecasting.

In addition, the proposed method, DBN with SGA, yielded the highest prediction (E1 measurement) comparing to all previous studies such as MLP with BP, the best prediction of CATS competition IJCNN'04 [4], the conventional DBNs with BP [9, 11], and hybrid models [13]. The details are shown in **Table 1**.

The meta-parameters obtained by random search method are shown in **Table 2**. And we found that the MSE of learning, i.e., given by one-ahead prediction results, showed that the proposed method has worse convergence compared to the conventional BP training. In **Figure 8**, the case of the first block learning MSE of two methods is shown. The convergence of MSE given by BP converged in a long training process and SGA gave unstable MSE of prediction. However, as the basic consideration of a sparse model, the better results of long-term prediction of the proposed method may successfully avoid the over-fitting problem which is caused by the model that is built too strictly by the training sample and loses its robustness for unknown data.

3.2 Real time series data

Three types of natural phenomenon time series data provided by Aalto University [17] were used in the one-ahead forecasting experiments of real time series data.

Method	E_1
DBN(SGA) [18]	170
DBN(BP) + ARIMA [14]	244
DBN [11] (BP)	257
Kalman Smoother (the best of IJCNN '04) [4]	408
DBN [9] (2 RBMs)	1215
MLP [9]	1245
A hierarchical Bayesian learning (the worst of IJCNN '04) [4]	1247
ARIMA [1]	1715
ARIMA+MLP(BP) [12]	2153
ARIMA+DBN(BP) [14]	2266

Table 1.
The long-term forecasting error comparison of different methods using CATS data.

	DBN with SGA	DBN with BP
The number of RBMs	3	1
Learning rate of RBM	0.048-0.055-0.026	0.042
Structure of DBN (the number of units and layers)	14-14-18-19-18-2	5-11-2-1
Learning rate of SGA or BP	0.090	0.090
Discount factor γ	0.082	—
Coefficient β	1.320	—

Table 2.
Meta-parameters of DBN used for the CATS data (block 1).

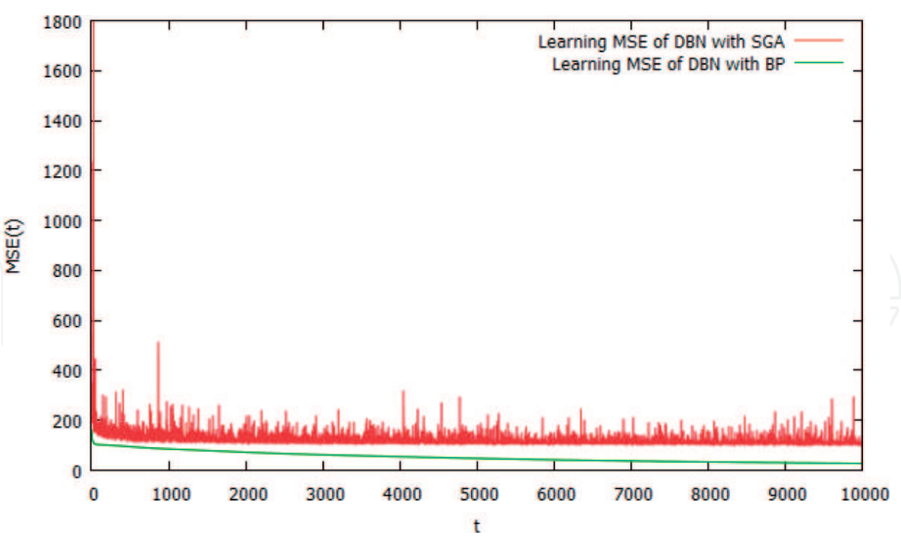


Figure 8.
Change of the learning error during fine-tuning (CATS data [1–980]).

- CO_2: Atmospheric CO_2 from continuous air samples weekly averages atmospheric CO_2 concentration derived from continuous air samples, Hawaii, 2225 data

- Sea level pressures: Monthly values of the Darwin sea level pressure series, A.D. 1882–1998, 1300 data

- Sunspot number: Monthly averages of sunspot numbers from A.D. 1749 to the present 3078 values

The prediction results of these three datasets are shown in **Figure 9**. Short-term prediction error is shown in **Table 3**. DBN with the SGA learning method showed its priority in all cases.

The efficiency of random search to find the optimal meta-parameters, i.e., the structure of RBM and MLP, learning rates, discount factor, etc. which are explained in Section 2.5 is shown in **Figure 10** in the case of DBN with SGA learning algorithm. The random search results are shown in **Table 4**.

We also used seven types of natural phenomenon time series data of TSDL [18]. The data to be predicted was chosen based on [19] which are named as Lynx, Sunspots, River flow, Vehicles, RGNP, Wine, and Airline. The short-term (one-ahead) prediction results are shown in **Figure 11** and **Table 5**.

From **Table 5**, it can be confirmed that SGA showed its priority to BP except the cases of Vehicles and Wine. From **Table 6**, an interesting result of random search for meta-parameter showed that the structures of DBN for different datasets were different, not only the number of units on each layer but also the number of RBMs. In the case of SGA learning method, the number of layer for Sunspots, River flow, and Wine were more than DBN using BP learning.

4. Discussions

The experiment results showed the DBN composed by multiple RBMs and MLP is the state-of-the-art predictor comparing to all conventional methods in the case of CATS data. Furthermore, the training method for DBN may be more efficient by the RL method SGA for real time series data than using the conventional BP algorithm. Here let us glance back at the development of this useful deep learning method.

- Why the DBN composed by multiple RBMs and MLP [11, 13] is better than the DBN with multiple RBMs only [9]?

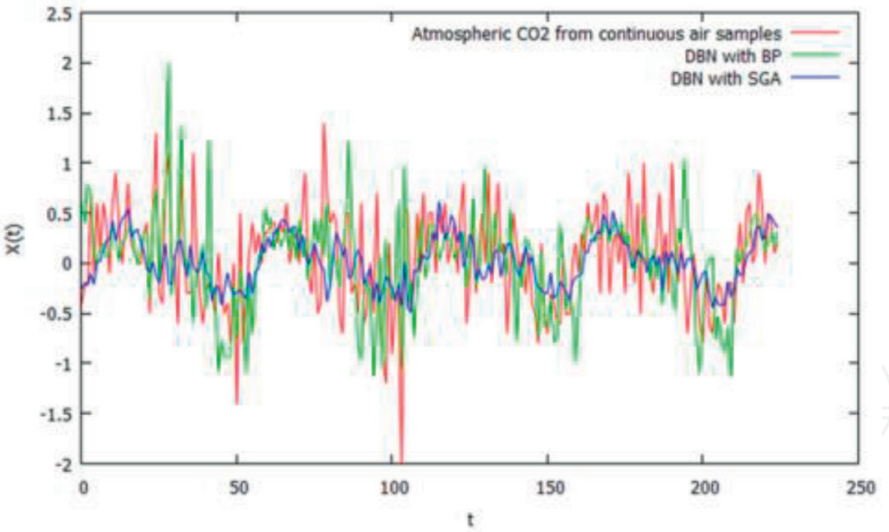
The output of the last RBM of DBN, a hidden unit of the last RBM in DBN, has a binary value during pretraining process. So the weights of connections between the unit and units of the visible layer of the last RBM are affected and with lower complexity than using multiple units with continuous values, i.e., MLP, or so-called full connections in deep learning architecture.

- How are RL methods active at ANN training?

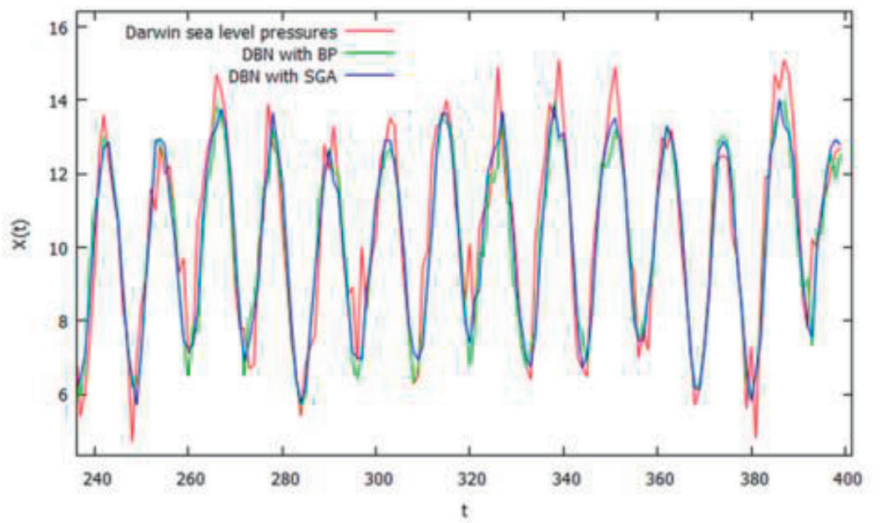
In 1992, Williams proposed to adopt a RL method named REINFORCE to modify artificial neural networks [8]. In 2008, Kuremoto et al. showed the RL method SGA is more efficient than the conventional BP method in the case of time series forecasting [6]. Recently, researchers in DeepMind Ltd. adopted RL into deep neural networks and resulted a famous game software AlphaGo [20–23].

- Why SGA is more efficient than BP?

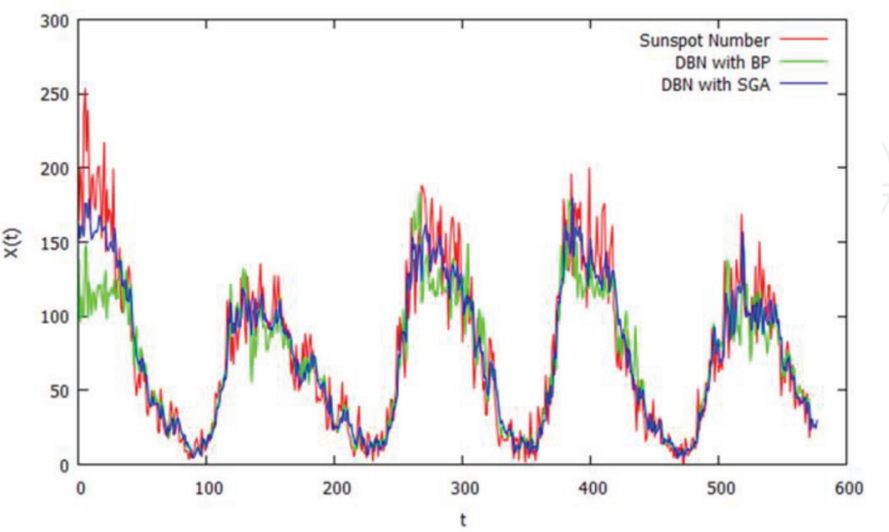
Generally, the training process for ANN by BP uses mean square error as loss function. So every sample data affects the learning process and results including noise data. Meanwhile, SGA uses reward which may be an error zone to modify the



(a)



(b)



(c)

Figure 9.
Prediction results by DBN with BP and SGA. (a) Prediction result of CO_2 data. (b) Prediction result of Sea level pressure data. (c) Prediction result of Sun spot number data.

Data	DBN with BP	DBN with SGA
CO2	0.2671	0.2047
Sea level pressure	0.9902	0.9003
Sun spot number	733.51	364.05

Table 3.
Prediction MSE of real time series data [17].

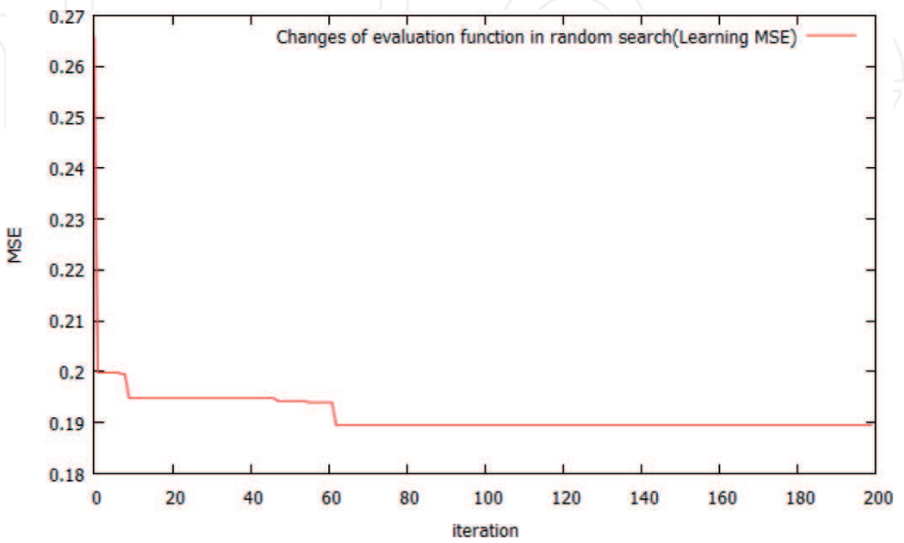


Figure 10.
Changes of learning error by random search for DBN with SGA.

Data series	Total data	Testing data	DBN with BP (the number of units)	DBN with SGA (the number of units)
CO_2	2225	225	15-17-17-1	20-18-7-2
Sea level pressure	1400	400	16-18-18-1	16-20-8-7-2
Sun spot number	3078	578	20-20-17-18-1	19-19-20-10-2

Table 4.
Meta-parameters of DBN used for real time series forecasting.

parameters of model. So it has higher robustness for the noisy data and unknown data for real problems.

5. Conclusions

A deep belief net (DBN) composed by multiple restricted Boltzmann machines (RBMs) and multilayer perceptron (MLP) for time series forecasting were introduced in this chapter. The training method of DBN is also discussed as well as a reinforcement learning (RL) method; stochastic gradient ascent (SGA) showed its priority to the conventional error back-propagation (BP) learning method. The robustness of SGA comes from the utilization of relaxed prediction error during the

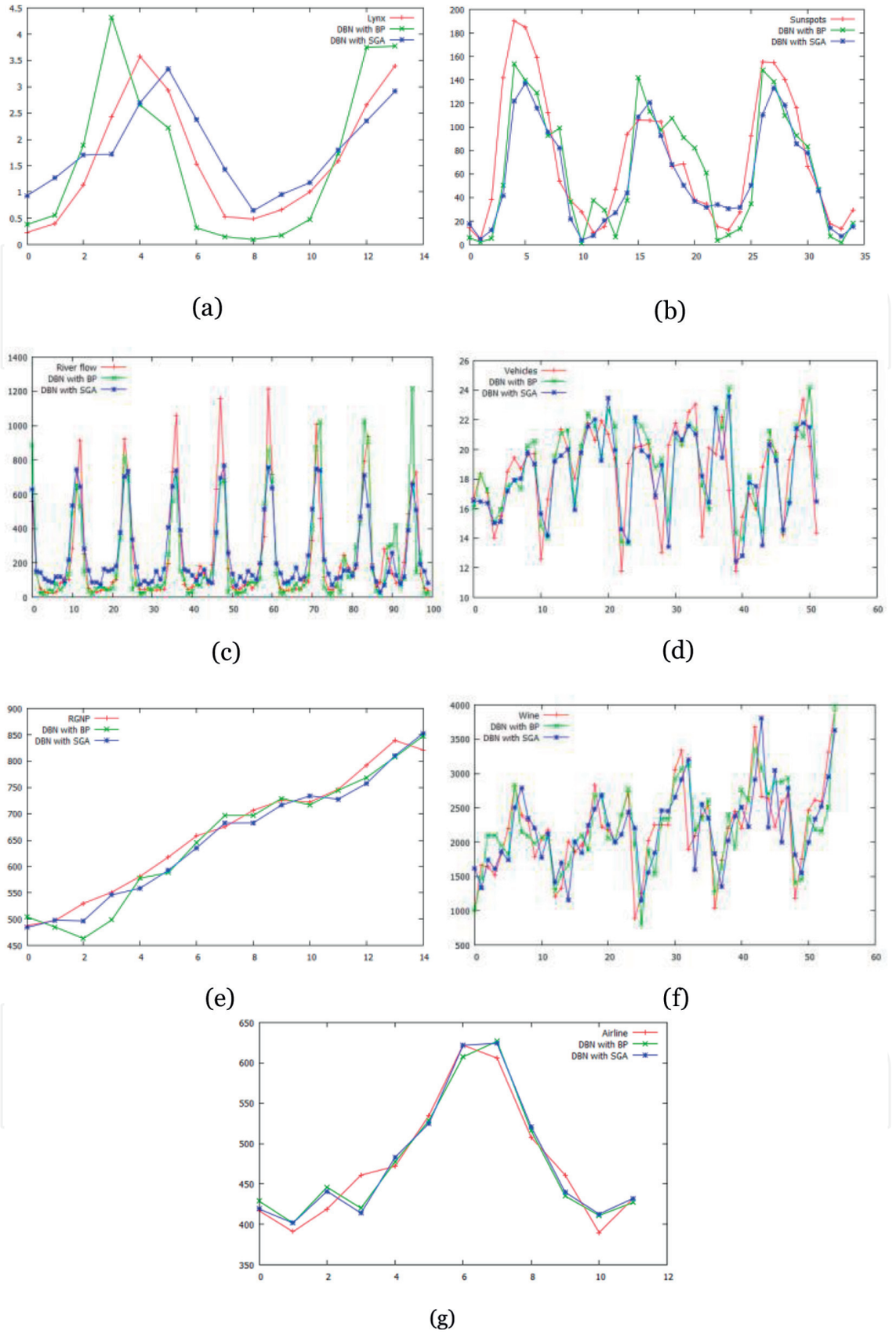


Figure 11.
Prediction results of natural phenomenon time series data of TSDL. (a) Prediction result of Lynx; (b) prediction result of sunspots; (c) prediction result of river flow; (d) prediction result of vehicles; (e) prediction result of RGNP; (f) prediction result of wine; and (g) prediction result of airline.

learning process, i.e., different from the BP method which adopts all errors of every sample to modify the model. Additionally, the optimization of the structure of DBN was realized by random search method. Time series forecasting experiments used

Data	DBN with BP	DBN with SGA
Lynx	0.6547	**0.3593**
Sunspots	999.54	**904.35**
River flow	24262.24	**16980.46**
Vehicles	**6.0670**	6.1919
RGNP	771.79	**469.72**
Wine	**138743.80**	224432.02
Airline	380.60	**375.25**

Table 5.
Prediction MSE of time series data of TSDL.

Series	Total data	Testing data	DBN with BP	DBN with SGA
Lynx	114	14	19-16-1	7-14-2
Sunspots	288	35	20-18-11-1	10-12-12-17-2
River flow	600	100	20-17-18-1	19-20-5-18-5-2
Vehicles	252	52	20-13-20-1	20-11-5-2
RGNP	85	15	18-20-1	19-15-2
Wine	187	55	16-15-12-1	18-12-13-11-2
Airline	144	12	15-4-1	13-7-2

Table 6.
Size of time series data and structure of prediction network.

benchmark CATS data, and real time series datasets showed the effectiveness of the DBN. As for the future work, there are still some problems that need to be solved such as how to design the variable learning rate and reward which influence the learning performance strongly and how to prevent the explosion of characteristic eligibility trace in SGA.

Author details

Takashi Kuremoto[1*], Takaomi Hirata[1], Masanao Obayashi[1], Shingo Mabu[1] and Kunikazu Kobayashi[2]

1 Yamaguchi University, Ube, Japan

2 Aichi Prefectural University, Nagakute, Japan

*Address all correspondence to: wu@yamaguchi-u.ac.jp

IntechOpen

© 2019 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. CC BY

References

[1] Box GEP, Pierce DA. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. Journal of the American Statistical Association. 1970;**65**(332):1509-1526

[2] Casdagli M. Nonlinear prediction of chaotic time series. Physica D. 1989;**35**: 335-356

[3] Lendasse A, Oja E, Simula O, Verleysen M. Time series prediction competition: The CATS benchmark. In: Proceedings of International Joint Conference on Neural Networks (IJCNN'04); 2004. pp. 1615-1620

[4] Lendasse A, Oja E, Simula O, Verleysen M. Time series prediction competition: The CATS benchmark. Neurocomputing. 2007;**70**:2325-2329

[5] NN3. http://www.neural-forecasting-competition.com/NN3/index.htm

[6] Rumelhart DE, Hinton GE, Williams RJ. Learning representation by back-propagating errors. Nature. 1986; **232**(9):533-536

[7] Kuremoto T, Obayashi M, Kobayashi M. Neural forecasting systems, Chapter 1. In: Weber C, Elshaw M, Mayer NM, editors. Reinforcement Learning, Theory and Applications. Rijeka, Croatia: InTech; 2008. pp. 1-20

[8] Kimura H, Kobayashi S. Reinforcement learning for continuous action using stochastic gradient ascent. In: Proceedings of 5th Intelligent Autonomous Systems (IAS-5); 1998. pp. 288-295

[9] Williams RJ. Simple statistical gradient following algorithms for connectionist reinforcement learning. Machine Learning. 1992;**8**:229-256

[10] Kuremoto T, Kimura S, Kobayashi K, Obayashi M. Time series forecasting using a deep belief network with restricted Boltzmann machines. Neurocomputing. Aug. 2014;**137**(5): 47-56

[11] Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science. 2006; **313**:504-507

[12] Kuremoto T, Hirata T, Obayashi M, Mabu S, Kobayashi K. Forecast chaotic time series data by DBNs. In: Proceedings of the 7th International Congress on Image and Signal Processing (CISP 2014); Oct. 2014. pp. 1304-1309

[13] Zhang GP. Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing. 2003; **50**:159-175

[14] Hirata T, Kuremoto T, Obayashi M, Mabu S. Time series prediction using DBN and ARIMA. In: Proceedings of International Conference on Computer Application Technologies (CCATS 2015). Matsue, Japan; Sep. 2015. pp. 24-29

[15] Hirata T, Kuremoto T, Obayashi M, Mabu S, Kobayashi K. Deep belief network using reinforcement learning and its applications to time series forecasting. In: Proceedings of International Conference on Neural Information Processing, (ICONIP'16), Lecture Notes in Computer Science (LNCS). Heidelberg, Germany: Springer. Vol. 9949. Kyoto, Japan; Oct. 18–21, 2016. pp. 30-37

[16] Hirata T, Kuremoto T, Obayashi M, Mabu S, Kobayashi K. Forecasting real time series data using deep belief net and reinforcement learning. Journal of Robotics, Network and Artificial Life.

2018;**4**(4):260-264. DOI: 10.2991/jrnal.2018.4.4.1

[17] Bergstra J, Bengio Y. Random search for hyper-parameter optimization. Journal of Machine Learning Research. 2012;**13**:281-305

[18] Aalto University Applications of Machine Learning Group Datasets. Available online at url: <http://research.ics.aalto.fi/eiml/datasets.shtml> (01-01-17)

[19] Hyndman RJ. Time Series Data Library (TSDL). 2013. Available online at url: ⟨http://robjhyndman.com/TSDL/⟩ (01-01-13)

[20] Adhikari R. A neural network based linear ensemble framework for time series forecasting. Neurocomputing. 2015;**157**:231-242

[21] Mnih V et al. Human-level control through deep reinforcement learning. Nature. 2015;**518**:529-533

[22] Sivler D et al. Mastering the game of go with deep neural networks and tree search. Nature. 2017;**529**:484-489

[23] Sivler D et al. Mastering the game of go without human knowledge. Nature. 2017;**550**:354-359