A Fast Incremental Learning Algorithm for Feed-forward Neural Networks Using Resilient Propagation

Joseph, Annie anak

Ozawa, Seiichi

(Citation)
Memoirs of the Graduate Schools of Engineering and System Informatics Kobe University,6:13-17

(Issue Date)
2014

(Resource Type)
departmental bulletin paper

(Version)
Version of Record

(URL)
https://hdl.handle.net/20.500.14094/81008841

Memoirs of the Graduate Schools of Engineering and System Informatics Kobe University
No. 6, pp. 13–17, 2014. doi:10.5047/gseku.e.2014.002

A Fast Incremental Learning Algorithm for Feed-forward Neural Networks Using Resilient Propagation

Annie anak Joseph and Seiichi Ozawa

Department of Electrical and Electronic Engineering, Graduate School of Engineering, Kobe University

(Received May 27, 2014; Accepted June 24, 2014; Online published June 30, 2014)

Keywords: Incremental Learning, Online classifier, gradient descent, Feed-forward Neural Network, Back-propagation

Fast learning under incremental learning environments is very important in real situations as data are generated rapidly over time. However, the input-output relationships that are trained before tend to be destroyed when new data are received. Therefore, the information on the previous data tends to be lost when the data are drawn from the different data distribution. This phenomenon is called "interference" or catastrophic forgetting. To solve this problem, Resource Allocating Network with Long Term Memory (RAN-LTM) is proposed by Kobayashi et al. in order to suppress the interference. In the original RAN-LTM, both new training data and memory items are trained based on the gradient descent method. However, gradient descent method usually leads to the slow learning even for simple problems. On the other hand, Resilient Back-propagation (R-prop) performs a direct adaptation for the step size of the weight update based on local gradient information. The principal idea of R-prop is that the signs of two consecutive partial derivatives are used to determine how connection weights are updated. When the signs of two consecutive partial derivatives coincide with each other, the update-value is increased in order to accelerate the learning in the shallow regions. In contrast, when the signs of two consecutive partial derivatives are changed, the update value is decreased by a decrease value. By conducting these procedures, the number of learning steps is significantly reduced compared to the original gradient descent method. Considering the advantages of R-prop, we propose a fast incremental learning algorithm for feed-forward neural network where the gradient descent method in RAN-LTM is accelerated based on R-prop. The performance of the proposed method is evaluated for several data sets and the results demonstrated that learning time of the extended version of RAN-LTM is greatly reduced compared to the original RAN-LTM.

1. Introduction

Incremental learning has become an important topic in machine learning nowadays since it is more applicable in practical situations such as computer security, intelligent user interfaces and so forth. In the incremental learning environments, the data is received either one by one or chunk by chunk (i.e., multiple data in one chunk). After the training, the data is discarded and only some important information is stored in the system. We often call this type of learning as "online" or "incremental learning" [1].

Since the incremental learning has been shown to be very helpful for a growing number of real world applications, most of the learning systems have been extended so that they can learn the generated data incrementally [2)-8)]. However, it creates another issue when new training data are drawn from a biased distribution. In the learning of neural network, interference happens when the input-output relationships that are trained previously tend to be collapsed (i.e., forgetting) due to the excessive adaptation of the connection weights for new data. Therefore, suppressing the interference is important in the incremental learning environments. In order to handle this issue in neural networks, Kobayashi et al. [9)] proposed an incremental learning system called Resource Allocating Network with Long Term Memory (RAN-LTM). This method is an extended version of Resource Allocating Network [9)] because the original RAN does not have an ability to suppress the interference (i.e., forgetting). The data with inputs and outputs of the networks which is accurately approximated are allocated into RAN-LTM. Therefore, when new data is received, not only a new training data but data that are stored in LTM are learned based on the gradient descent method. The main issue of the gradient descent method is that the learning tends to be slow even for a simple problem. To solve this problem, we propose a new version of RAN-LTM that aims to accelerate the learning time.

R-prop stands for resilient back-propagation that performs a first-order learning algorithm for neural network. In our proposed method, RAN-LTM is extended such that R-prop is used to accelerate the learning in the original RAN-LTM. The main purpose of R-prop is to overcome the slow learning in gradient descent method. In the learning of R-prop, step size of the weight-update according to the error function is performed. The major difference between R-prop with the other adaptive techniques is that the step size of weight update is determined by the signs of two consecutive partial derivatives instead of monitor the magnitude of the gradients.

If the signs of the two consecutive gradients remained the same, the step size is increased by some constant value called "increase factor" in order to accelerate the convergence in shallow regions. On the other hand, when the signs of two consecutive partial derivatives are changed, it implies that the last update is too big; hence, the update-value is decreased with decrease factor. In our proposed method, only the connection weights and the network biases are learned based on R-prop technique.

The rest of this paper is organized as follows: Section 2 briefly explains RAN-LTM and a new version of RAN-LTM based on R-prop is proposed in Section 3. Then, Section 4 demonstrates the performance of our proposed method for several data sets. Finally concluding remarks and future work are mentioned in Section 5.

2. Resource Allocating Network-Long Term Memory (RAN-LTM)

RAN-LTM consists of two major parts: Resource Allocating Network (RAN) and Long Term Memory (LTM), which learns connection weights incrementally and the RBF units is added automatically based on RAN.

2. 1 Resource Allocating Network (RAN)

RAN is an extended version of Radial Basis Function (RBF) networks [10] which was originally proposed by Platt [11]. In the learning of RAN, the initial number of hidden nodes is set to one at the beginning of the learning stage; hence, RAN possesses simple approximation ability at first. The approximation ability of RAN is developed by adding new hidden nodes when new training data is received continuously. Figure 1 illustrates the network architecture of RAN.

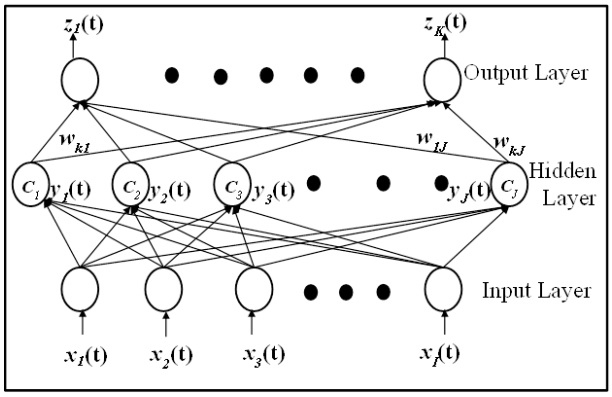

Fig. 1 Network architecture of Resource Allocating Network (RAN).

In RAN, when a new input data $\mathbf{x}_p = \{x_{p1}, \cdots, x_{pI}\}^t$ is given, the RBF output $\mathbf{y}_p = \{y_{p1}, \cdots, y_{pJ}\}^t$ is calculated based on the distance between the *p*th input and center vector of the *j*th hidden unit $\mathbf{c}_j = \{c_{j1}, \cdots, c_{jI}\}^t$ as follows:

$$y_{pj} = \exp\left(\frac{(\|\mathbf{x}_p - \mathbf{c}_j\|)^2}{2\sigma_j{}^2}\right) \quad (j = 1, \dots, J) \tag{1}$$

where *I* and *J* are the number of input units and hidden units, respectively. σ_j^2 is a variance of the *j*th radial basis. The network outputs $\mathbf{z}_p = \{z_{p1}, \cdots, z_{pK}\}^t$ are calculated by

$$z_{pk} = \sum_{j=1}^{J} w_{kj} y_{pj} + \gamma_k \quad (k = 1, \dots, K) \tag{2}$$

where w_{kj} is a connection weight from the *j*th hidden unit to the *k*th output unit, and γ_k is a bias of the *k*th output unit. Here, *K* is the number of output units. The network output is calculated by Eqs. (1) - (2) when a new training data is received and the error $\boldsymbol{E}_p$ between the output $\mathbf{z}_p$ and target $\mathbf{T}_p$ for the *p*th input is evaluated as follows:

$$E_p = \sqrt{\sum_{k=1}^{K} (T_{pk} - z_{pk})^2}\,. \tag{3}$$

Next, either one of the following procedure is carried out depending on which condition is satisfied.

(1) First condition:

If $E_p > \varepsilon$ and $\|\mathbf{x}_p - \mathbf{c}^*\| > \delta\ (t)$
A hidden unit is added (i.e., $J \leftarrow J+1$) if an error is larger than the positive constant value ε and the distance between the *p*th input $\mathbf{x}_p$ and its nearest center vector $\mathbf{c}^*$ is larger than a positive value $\delta(t)$. Then, the center vector $\mathbf{c}_J$, connection weights w_{kj}, and variance σ_J^2 (network parameters for the *J*th hidden node) are set as follows:

$$c_{Ji} = x_{pi} \quad (i = 1, \dots, I) \tag{4}$$

$$w_{kJ} = T_{pk} - z_{pk} \tag{5}$$

$$\sigma_J = \kappa\|\mathbf{x}_p - \mathbf{c}^*\| \tag{6}$$

where κ is a positive constant and $\delta(t)$ is decreased with time *t* as follows:

$$\delta(t) = \max\left[\delta_{max}\exp\left(\frac{t}{\tau}\right), \delta_{min}\right] > 0 \tag{7}$$

where δ_{max} and δ_{min} are maximum and minimum values of $\delta(t)$, respectively. In addition, τ is a decay constant.

(2) Second Condition:

If the first condition is not met, the following parameters are updated based on the gradient descent method.

$$w_{kj}{}^{NEW} = w_{kj}{}^{OLD} + \alpha(T_{pk} - z_{pk})y_j \tag{8}$$

$$c_{ji}{}^{NEW} = c_{ji}{}^{OLD} + \frac{\alpha}{\sigma_j{}^2}(x_{pi} - c_{ji})y_j \sum_k (T_{pk} - z_{pk})w_{kj} \tag{9}$$

$$\gamma_k{}^{NEW} = \gamma_k{}^{OLD} + \alpha(T_{pk} - z_{pk}) \tag{10}$$

where α is a positive learning ratio.

In the learning of RAN, it does not have an ability to suppress the interference. Therefore, RAN-LTM was proposed by Kobayashi et al. [9] in order to solve the interference problem in RAN.

2.2 Resource Allocating Network with Long Term Memory (RAN-LTM)

In RAN-LTM, memory-based learning approach is applied to the original RAN. The data stored in LTM is called "LTM data". These data are used to suppress the interference.

Figure 2 shows the architecture of an extended version of RAN. The extended RAN consists of two major parts: RAN and an external memory called *Long-Term Memory* (LTM). The extended version of learning approach is called Resource Allocating Network with Long-Term Memory (RAN-LTM) [9]. Representative input-output pairs are retrieved from the mapping function obtained in RAN and they are stored in LTM. These pairs are called *memory items* and some of them are retrieved from LTM to learn together with a newly received

training data. The learning of new training data with memory items could prevent RAN from the mapping function acquired previously to be lost even though the data is given incrementally.

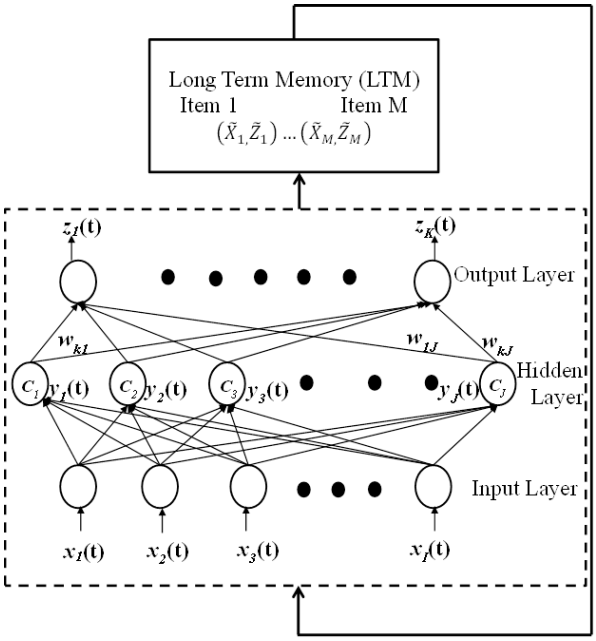

Fig. 2 Architecture of RAN-LTM.

In RAN-LTM, only some data are selected based on the effectiveness of the data to be added into LTM. To do this, we select the nearest memory items to the centers for actives hidden nodes to be added into LTM. When a new data is given to RAN, the memory items are retrieved from the LTM and the outputs of the hidden nodes are calculated based on the distance between memory items and the newly received data. If the output of hidden nodes is greater than the threshold, the memory item is retrieved and learned together with newly added data to suppress the forgetting.

3. Extended RAN-LTM-Rprop

This section describes an extended version of RAN-LTM. As stated in Section 1, if the first condition is not satisfied, the second process is carried out in order to update the learning in RAN-LTM; that is, the network parameters (Eqs. (8) – (10)) are updated based on the traditional gradient descent method. Considering that the training using the gradient descent method is slow, more efficient way is needed in order to accelerate the convergence of RAN-LTM. To overcome this problem, we have extended the original RAN-LTM such that R-prop is applied to update the network weights and biases in RAN-LTM. For the notational simplicity, the extended version of RAN-LTM is denoted as RAN-LTM-Rprop. The basic concept of R-prop is first described in order to further discuss about the extended version of RAN-LTM.

3.1 Resilient Back-propagation (R-prop)

R-prop stands for "Resilient Back-propagation" which was originally proposed by Riedmiller et al. [12] in order to accelerate the training in a feed-forward neural network. Before the further explanation of the basic idea of R-prop, let us denote the connection weight from neuron i to neuron j in a neural network as w_{ij} and E be an error that is differentiable with respect to the connection weights.

In the original back-propagation neural network, the connection weights are tuned such that the network can learn a desired mapping function between inputs and outputs. The partial derivative $\frac{\partial E}{\partial w_{ij}}$ is computed repeatedly for each connection weight in a network in order to obtain the minimum error between the actual outputs and the desired outputs. However, tuning the connection weights by the gradient descent method usually leads to slow convergence especially in the shallow regions. Therefore, many adaptive techniques have been proposed in order to solve the slow convergence in the gradient descent method. These adaptive techniques included both global and local adaptive techniques [13]. One of them is R-prop and it is a local adaptive technique. Different from the other adaptive techniques, the basic idea of R-prop is to adapt the connection weights by observing only the signs of the two consecutive partial derivatives in order to indicate the direction of the step size [12]. Here, the size of the weight change is determined by the step size or update value Δ_{ij} as follows:

$$\Delta w_{ij}(t) = \begin{cases} -\Delta_{ij}(t), & \text{if } \frac{\partial E}{\partial w_{ij}} > 0 \\ +\Delta_{ij}(t), & \text{if } \frac{\partial E}{\partial w_{ij}} < 0 \\ 0, & \text{else} \end{cases} . \quad (11)$$

Then, the next step is to determine the new update value $\Delta_{ij}(t)$. This can be done by observing the signs of local gradients; that is, the signs of two consecutive partial derivatives are observed as shown in the following rule:

$$\Delta_{ij}{}^{(t)} = \begin{cases} \eta^+ * \Delta_{ij}{}^{(t-1)}, & \text{if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} * \frac{\partial E}{\partial w_{ij}} > 0 \\ \eta^- * \Delta_{ij}{}^{(t-1)}, & \text{if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} * \frac{\partial E}{\partial w_{ij}} < 0 \\ \Delta_{ij}{}^{(t-1)}, & \text{else} \end{cases} . \quad (12)$$

If the two consecutive partial derivatives possess the same sign, the update value is increased by an increase factor, whereas if the two consecutive partial derivative changes its sign, the update value is decreased by a decrease factor. The values of increase and decrease factor are suggested to be within the range of $0 < \eta^- < 1 < \eta^+$ [14] where the value of η^+ and η^- are suggested to be 1.2 and 0.5, respectively. These values are based on both the empirical evaluations and theoretical considerations [14].

The previous works had shown that the effectiveness of the R-prop and it can converge faster than the ordinary gradient descent algorithm [15].

3.2 Learning Environments of Extended RAN-LTM

When an input $\mathbf{x}_p$ is given to RAN, the network output is calculated by Eqs. (1) - (2) and the error E_p between the output $\mathbf{z}_p$ and target $\mathbf{T}_p$ for the pth input is calculated by Eq. (3). If $E_p > \varepsilon$ and $\|\mathbf{x}_p - \mathbf{c}^*\| > \delta(t)$, new hidden node is added into RAN-LTM. Otherwise, the network parameters need to be updated based on the new data and some memory items from LTM. Here, the memory items from LTM are recalled in order to suppress the interference. In RAN-LTM, the connection weight w_{kj} and the bias γ_k are learned based on R-prop

technique. The connection weights and the biases are updated by the following rule:

$$w_{kj}{}^{NEW} = w_{kj}{}^{OLD} + \Delta w_{kj}(t) \quad (13)$$

$$\gamma_k{}^{NEW} = \gamma_k{}^{OLD} + \Delta\gamma_k\,(t) \quad (14)$$

where

$$\Delta w_{kj}(t) = \begin{cases} -\Delta_{kj}^{w}(t)\,, & \text{if } (T_{pk} - z_{pk})y_j > 0 \\ +\Delta_{kj}^{w}(t)\,, & \text{if } (T_{pk} - z_{pk})y_j < 0 \\ 0\,, & \text{else} \end{cases} \quad (15)$$

and

$$\Delta\gamma_k\,(t) = \begin{cases} -\Delta_{k}^{\gamma}(t)\,, & \text{if } T_{pk} - z_{pk} > 0 \\ +\Delta_{k}^{\gamma}(t)\,, & \text{if } T_{pk} - z_{pk} < 0\,. \\ 0\,, & \text{else} \end{cases} \quad (16)$$

Here, $\frac{\partial E}{\partial w_{kj}} = (T_{pk} - z_{pk})y_j$ and $\frac{\partial E}{\partial \gamma_k} = T_{pk} - z_{pk}$. Then, the next step is to determine the new update values $\Delta_{kj}^{w}(t)$ and $\Delta_{k}^{\gamma}(t)$. This can be done by observing the sign of two consecutive partial derivatives as shown in the following rule:

$$\Delta_{kj}^{w}(t) = \begin{cases} \eta^{+} * \Delta_{kj}^{w}(t-1)\,, & \text{if } \frac{\partial E}{\partial w_{kj}}(t-1) * \frac{\partial E}{\partial w_{kj}}(t) > 0 \\ \eta^{-} * \Delta_{kj}^{w}(t-1)\,, & \text{if } \frac{\partial E}{\partial w_{kj}}(t-1) * \frac{\partial E}{\partial w_{kj}}(t) < 0 \\ \Delta_{kj}^{w}(t-1)\,, & \text{else} \end{cases} \quad (17)$$

$$\Delta_{k}^{\gamma}(t) = \begin{cases} \eta^{+} * \Delta_{k}^{\gamma}(t-1)\,, & \text{if } \frac{\partial E}{\partial \gamma_k}(t-1) * \frac{\partial E}{\partial \gamma_k}(t) > 0 \\ \eta^{-} * \Delta_{k}^{\gamma}(t-1)\,, & \text{if } \frac{\partial E}{\partial \gamma_k}(t-1) * \frac{\partial E}{\partial \gamma_k}(t) < 0\,. \\ \Delta_{k}^{\gamma}(t-1)\,, & \text{else} \end{cases} \quad (18)$$

If the signs of two consecutive partial derivatives remained the same, the update value is increased by an increase factor while its signs are changed; the update value is decreased by a decrease factor. The value of increase and decrease factor is set to 1.2 and 0.5, respectively. The learning of RAN-LTM-Rprop is shown in Algorithm 1.

4. Performance Evaluation

To evaluate the performance of the proposed method, three data sets (Ozone, Banknote identification and Spambase) are selected from the Machine Learning Repository [16]. Table 1 shows the summary of the three evaluated data sets. The network parameters, σ_j^2 and $\delta\,(t)$ for Ozone data is set to 50 and 0.6, respectively while for Banknote identification data is set to 2 and 0.01, respectively. In addition, the value for σ_j^2 and $\delta\,(t)$ is set to 10 and 0.2, respectively for Spambase data.

The CPU of the computer used for the performance evaluation has a following specification:

Algorithm 1: RAN-LTM-Rprop

- **Require**: Training data $\mathbf{x}_p = \{x_{p1}, \ldots, x_{pI}\}^t$ and J =1.

1) Calculate RBF outputs $\boldsymbol{y}_p$ based on Eq. (1).
2) Calculate the network output z_{pk} by Eq. (2).
3) Measure E_p between the output $\mathbf{z}_p$ and target $\mathbf{T}_p$ for the pth input by Eq. (3).
4) If $E_p > \varepsilon$ *and* $\|\boldsymbol{x}_p - \boldsymbol{c}^*\| > \delta\,(t)$
5) $J \leftarrow J + 1$
6) Set the center vector $\mathbf{c}_J$, connection weights w_{kj}, and variance σ_J^2 (network parameters for the Jth hidden node).
7) else
8) Call some memory items from LTM.
9) Update the connection weights and the biases by using Eqs. (13) - (18) for $\mathbf{x}_p$ and some memory items from LTM.

1. Intel(R) Core(TM)2 Duo (3.16 GHz) with 2GB of random access memory.
2. The program development and the experiments are carried out with Matlab (R2007b).
3. Windows 7 (32-bit OS).

The performance of incremental learning depends on the sequence of training data. Hence, we conduct 25 trials with different training sequences and the results are averaged over the training sequences.

4.1 Effectiveness of RAN-LTM-Rprop

In order to evaluate the effectiveness of the proposed method, the learning accuracy and learning time of RAN-LTM-Rprop are compared with the previous version of RAN-LTM. The training of connection weights in RAN-LTM is conducted based on the gradient descent method.

Table 2 (a) and (b) show the learning time [sec.] and recognition accuracy [%] for the three data sets. As seen in Table 2 (a), better performance is always achieved by the proposed method; that is, the learning time of RAN-LTM-Rprop is significantly reduced compared to the original RAN-LTM. When R-prop technique applies to the learning in the connection weights, the size of update value is determined by the signs of two consecutive partial derivatives. Therefore, when the signs of two consecutive partial derivatives are the same, the update value is increased by some increase factor. The increase factor is set to 1.2. By increasing the update value the learning could be faster compared to the traditional RAN-LTM that trains the connection weights using the traditional gradient descent method.

On the other hand, the recognition accuracies are almost similar between the proposed method and RAN- LTM for the three data sets. Although the learning time of RAN-LTM-Rprop is faster than the original RAN-LTM, the recognition accuracies are remained stable between both learning algorithms. This is because when the learning of the connection weights changes its sign, it implies that the last up-

Table 1 Evaluated Data Sets

Datasets	#Attrib	#Class	#Train	#Test
Ozone	72	2	900	924
Banknote	4	2	800	572
Spambase	57	2	980	1000

Table 2 Performance comparison results between the proposed RAN-LTM-Rprop and RAN-LTM

(a) Learning Time [sec.]

Datasets	RAN-LTM-Rprop	RAN-LTM
Ozone	1.73±0.06	12.96±5.69
Banknote	16.48±0.93	8685.43±2145.23
Spambase	10.37±0.08	75.08±0.06

(b) Recognition Accuracy [%]

Datasets	RAN-LTM-Rprop	RAN-LTM
Ozone	97.7±0.01	96.9±0.03
Banknote	95.7±0.07	93.8±0.12
Spambase	89.8±0.5	89.9±1.0

date is too big, and then the update value is decreased by some decrease factor. This step is very important to ensure the training is not oscillated. Here, the decrease value is set to 0.5.

However, the selection of network parameters would affect the recognition accuracy in both learning algorithms. Therefore, selection of appropriate network parameters is important to ensure the optimum performance of the proposed method.

5. Conclusions

In this paper, we propose a fast incremental learning algorithm for feed-forward neural networks in which an expected forgetting is effectively suppressed by extending Resource Allocating Network with Long Term Memory (RAN-LTM). In the proposed method, Resilient Back-propagation is introduced in the learning of connection weights and biases in RAN-LTM. The proposed method is called RAN-LTM-Rprop. In the original RAN-LTM, the connection weights, the biases and the RBF centers are learned based on the conventional gradient descent method. Therefore, the learning is usually slow. We extend the original RAN-LTM in which R-prop is used to learn the connection weight as well as the network biases.

To evaluate the performance of the extended version of RAN-LTM, three data sets (Ozone, Banknote identification and Spambase data) are selected from the UCI Machine Learning Repository [16]. The learning time and recognition accuracy are measured for both RAN-LTM and the proposed method. The experiment results demonstrate that the learning time of the extended RAN-LTM is greatly reduced compared to the original RAN-LTM while the recognition accuracy of the proposed method is almost similar to RAN-LTM for the three data sets. In the future, we plan to apply the extended version of RAN-LTM with more data sets including real world data sets.

References

1) Ozawa S, Pang S, and Kasabov N; "Incremental Learning of Chunk Data for On-line Pattern Classification Systems," IEEE Trans. on Neural Networks, Vol. 19, No. 6, pp. 1061-1074 (2008)

2) Oyama T, Karungaru S G, Tsuge S, Mitsukura Y, and Fukumi M; "Fast Incremental Algorithm of Simple Principal Component Analysis," IEEJ Trans EIS, 129:112-117 (2009)

3) Joseph A, Jang Y M, Ozawa S, and Lee M; "Extension of Incremental Linear Discriminant Analysis to Online Feature Extraction under Nonstationary Environments," In: Proc. of 19th Int. Conf. on Neural Information Processing, pp. 640-647 (2012)

4) Aoki D, Omori T, and Ozawa S; "A Robust Incremental Principal Component Analysis for Feature Extraction from Stream Data with Missing Values," In: Proc. Int. Joint Conf. on Neural Networks, pp. 1-8 (2013)

5) Jang Y M, Lee M, and Ozawa S; "A Real-Time Personal Authentication System Based on Incremental Feature Extraction and Classification of Audiovisual Information," Evolving System, 2:261-272 (2011)

6) Minku F L, Inoue H, and Yao X; "Negative Correlation in Incremental Learning," Natural Computing Journal Special Issue on Nature-Inspired Learning and Adaptive Systems, 8:289-320 (2009)

7) Elwell R and Polikar R; "Incremental Learning of Concept Drift in Nonstationary Environments," IEEE Trans. on Neural Network, 22:1517-1531 (2011)

8) Zhao H, Yuen P C, and Kwok J T; "A Novel Incremental Principal Component Analysis and Its Application for Face Recognition," IEEE Trans. on Systems, Man and Cybernetics, Part B, 36:873-886 (2006)

9) Kobayashi M, Zamani A, Ozawa S, and Abe S; "Reducing Computations in Incremental Learning for Feed-forward Neural Network with Long-Term Memory," Proc. Int. Joint. Conf. on Neural Networks, pp. 1989-1994 (2001)

10) Poggio T and Girosi F; "Networks for Approximation and Learning," Proc. IEEE. Trans. on Neural Network, 78(9), 1481-1497 (1990)

11) Platt, J C.; "A Resource-Allocating Network for Function Interpolation," Neural Computation, 3(2), 213-225, (1991)

12) Riedmiller M and Braun H; "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm," Proc. Int'l Conf. Neural Networks, pp. 586-591, (1993)

13) M. Riedmiller; "Advanced Supervised Learning in Multi-layer Perceptrons-from Backpropagation to Adaptive Learning Algorithms," Comput. Standards Interfaces, 16 (5) 265–278, (1994)

14) Riedmiller M; "RPROP: Description and Implementation Details," Technical Report, University of Karlsruhe, (1994)

15) Micheal B.B; "The Steepest Descent Method," Nonlinear Optimization with Engineering Applications, Springer Optimization and Its Applications, Vol. 19, pp 1-8 (2008)

16) Asunction S and Newman D. J; "UCI Machine Learning Repository.," UC. Irvine, School of Info. and Comp. Sci. (2007)