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”

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. 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. 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 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.

\[
E_p = \sum_{k=1}^{K} (T_{pk} - z_{pk})^2 .
\]  (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 \( \|x_p - c^*\| > \delta(t) \)

A hidden unit is added (i.e., \( J \leftarrow J + 1 \)) if an error is larger than the positive constant value \( \varepsilon \) and the distance between the \( p \)th input \( x_p \) and its nearest center vector \( c^* \) is larger than a positive value \( \delta(t) \). Then, the center vector \( c_{ji} \), connection weights \( w_{kj} \) and variance \( \sigma_j^2 \) (network parameters for the \( j \)th hidden node) are set as follows:

\[
c_{ji} = x_{pi} \quad (i = 1, ..., I)
\]  (4)

\[
w_{kj} = T_{pk} - z_{pk}
\]  (5)

\[
\sigma_j = \kappa \|x_p - c^*\|
\]  (6)

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

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

where \( \delta_{\text{max}} \) and \( \delta_{\text{min}} \) are maximum and minimum values of \( \delta(t) \), respectively. In addition, \( \tau \) 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.

\[
w_{kj}^{\text{NEW}} = w_{kj}^{\text{OLD}} + \alpha (T_{pk} - z_{pk})y_j
\]  (8)

\[
c_{ji}^{\text{NEW}} = c_{ji}^{\text{OLD}} + \frac{\alpha}{\sigma_j^2} (x_{pi} - c_{ji}) y_j \sum_{k} (T_{pk} - z_{pk}) w_{kj}
\]  (9)

\[
y_k^{\text{NEW}} = y_k^{\text{OLD}} + \alpha (T_{pk} - z_{pk})
\]  (10)

where \( \alpha \) 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.

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 update value is determined by the step size \( \eta \) and the error between the actual outputs and the desired outputs. 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 \(^{13}\). Here, the size of the weight change is determined by the step size or update value \( \Delta_{ij} \) 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:

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

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 \( 0 < \eta^+ < 1 < \eta^- \)

By using R-prop, the network output is calculated by Eqs. (1) - (2) and the error \( E_p \) between the output \( z_p \) and target \( T_p \) for the \( p \)-th input is calculated by Eq. (3). If \( E_p > \varepsilon \) and \( \| x_p - c \| > \delta \), a 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 \( w_{ij} \) and the bias \( \gamma_k \) are learned based on R-prop.

3.2 Learning Environments of Extended RAN-LTM

When an input \( x_p \) is given to RAN, the network output is calculated by Eqs. (1) - (2) and the error \( E_p \) between the output \( z_p \) and target \( T_p \) for the \( p \)-th input is calculated by Eq. (3). If \( E_p > \varepsilon \) and \( \| x_p - c \| > \delta \), a 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 \( w_{ij} \) and the bias \( \gamma_k \) are learned based on R-prop.
Algorithm 1: RAN-LTM-Rprop

- **Require:** Training data \( \mathbf{x}_p = \{x_{p1}, \ldots, x_{pk}\} \) and \( J = 1 \).

1. Calculate RBF outputs \( y_p \) based on Eq. (1).
2. Calculate the network output \( z_{pk} \) by Eq. (2).
3. Measure \( E_p \) between the output \( z_p \) and target \( T_p \) for the \( p \)th input by Eq. (3).
4. If \( E_p > \varepsilon \) and \( \| \mathbf{x}_p - \mathbf{c}^* \| > \delta \) (\( t \))
5. \( J \leftarrow J + 1 \)
6. Set the center vector \( \mathbf{c}_p \), connection weights \( w_{pk} \), and variance \( \sigma^2_{z_p} \) (network parameters for the \( J \)th 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).

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-
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) are selected from the UCI Machine Learning Repository [17]. 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.

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 [17]. 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


<table>
<thead>
<tr>
<th>Datasets</th>
<th>RAN-LTM-Rprop</th>
<th>RAN-LTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ozone</td>
<td>1.73±0.06</td>
<td>12.96±5.69</td>
</tr>
<tr>
<td>Banknote</td>
<td>16.48±0.93</td>
<td>8685.43±2145.23</td>
</tr>
<tr>
<td>Spambase</td>
<td>10.37±0.08</td>
<td>75.08±0.06</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Datasets</th>
<th>RAN-LTM-Rprop</th>
<th>RAN-LTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ozone</td>
<td>97.7±0.01</td>
<td>96.9±0.03</td>
</tr>
<tr>
<td>Banknote</td>
<td>95.7±0.07</td>
<td>93.8±0.12</td>
</tr>
<tr>
<td>Spambase</td>
<td>89.8±0.5</td>
<td>89.9±1.0</td>
</tr>
</tbody>
</table>