THREE TERM BACKPROPAGATION ALGORITHM FOR CLASSIFICATION PROBLEM

Siti Mariyam Shamsuddin, Fadhlina Izzah Saman∗, Maslina Darus†

Abstract: Standard Backpropagation Algorithm (BP) usually utilizes two term parameters; Learning Rate $\alpha$ and Momentum Factor $\beta$. Despite the general success of this algorithm, there are several drawbacks such as existence of local minima, slow rates of convergence and modification of algorithm requires complex computations. In this study, further analysis of proportional factor ($\gamma$) for 3-Term BP is investigated on various scales of datasets; small, medium and large. Experiments are conducted using three UCI dataset; Balloon, Iris and Cancer. The results show that the 3-Term BP outperforms standard BP for small scale data, but does not work well for medium and large scale dataset.

Key words: BP Algorithm, 3-Term BP, learning rate, momentum rate, proportional factor

Received: ??
Revised and accepted: ??

1. Introduction

Classification is defining as a set of groups according to their characteristics, in which each case is analyzed and classified into related groups. Classification can be used to understand the existing data and to predict how new instances will behave. The goal of data classification is to organize and categorize the data into distinct classes. A model is first created based on the data distribution, and will be used for classifying new data. Given this model, a class can be predicted for new data. Neural Network (NN) must be trained to classify certain data patterns to certain outputs.

∗Siti Mariyam Shamsuddin, Fadhlina Izzah Saman
Soft Computing Research Group, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, E-mail: mariyam@utm.my

†Maslina Darus
Mathematical Soft Computing Research Group, Faculty of Science and Technology, Universiti Kebangsaan Malaysia, E-mail: maslina@pkrisc.cc.ukm.my

©ICS AS CR 2007
2. Back propagation Algorithm (BP)

Backpropagation (BP) algorithm is a supervised learning method for multi-layered feedforward NN. It is commonly used for learning algorithm for training NN. The BP algorithm involves backward error correction of the network weights. Training is usually done by weights updating iteratively using a mean-square error function. Traditionally, the standard BP algorithm utilizes two terms parameters called Learning Rate ($\alpha$) and momentum factor ($\beta$) for weight adjustment.

Despite the general success of the BP algorithm, there are several drawbacks and limitations that still exist [1]. Limitations of standard BP include the existence of temporary, local minima resulting from the saturation behavior of the activation function. It also has slow convergence rates. It is relatively slow for network with more than one hidden layer. Some of the modification of BP algorithm requires complex and high computational function of a neuron, such as sigmoid function as shown below [4]

$$f(\text{net}_j) = \frac{1}{1 + e^{-\text{net}_j}}.$$ 

In the second phase, the errors calculated in the output layer are back propagated to the hidden layers where the synaptic weights are updated to reduce the error. This learning process is repeated until the output error value, for all patterns in the training set, are below a specified value.

The BP algorithm will change the current weights iteratively such that the system error function $E$ is minimized. This process is repeated iteratively until convergence is met. Typically, the error measure used in BP algorithm is the mean square error (MSE) [5]. The aim of the network learning is to minimize the error of the output signal by modifying the weights. The mean square error for $p^{th}$ output node is defined as:

$$E_p = \frac{1}{2} \sum_{j=1}^{N} (t_{pj} - o_{pj})^2,$$

where

- $E_p$ error for the $p^{th}$ presentation vector
- $t_{kj}$ is the desired value from output node ($k$) to hidden node ($j$),
- $o_{kj}$ is the network value from output node ($k$) to hidden node ($j$).

The weight adaptation in standard BP is defined as:

$$\Delta W_{kj}(n) = \alpha (t_k - o_k) o_k (1 - o_k) O_j + \beta \Delta W_{kj}(n - 1) \quad (1)$$

where

- $\alpha$ is the Learning Rate,
- $\beta$ is the Momentum Factor, and
- $\beta \Delta W_{kj}(n - 1)$ is proportional to the previous value of the incremental change of the weights.
From Equation (1), we can see that there are two terms added, namely, the $\alpha$ and $\beta$. These are the two terms that are usually utilized in standard BP or also known as two-term BP. The best choice of $\alpha$ depends on the problem, and normally it needs a trial and error process before a good choice is found. The larger the learning rate, the bigger the step and the faster the convergence [5]. However, if $\alpha$ value is made too large, the algorithm becomes unstable. On the other hand, if the $\alpha$ value is set to be too small, the algorithm will take quite a long time to converge, and will lead to slow convergence [6]. However, if the $\alpha$ value is too big, oscillation and overshooting of minimum will occur. The summary of $\alpha$ is shown in Tab. I.

<table>
<thead>
<tr>
<th>Value</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>• Slow convergence</td>
</tr>
<tr>
<td>Big</td>
<td>• Bigger steps</td>
</tr>
<tr>
<td></td>
<td>• Faster convergence</td>
</tr>
<tr>
<td></td>
<td>• Oscillation and overshooting of minimum if $\alpha$ value too big</td>
</tr>
</tbody>
</table>

Tab. I The behavior of $\alpha$.

Another possible way to improve the rate of convergence is by adding some momentum to the weight adjustment expression [7]. This can be accomplished by adding a fraction of the previous weight change to the current weight change. Parameter $\beta$ allows a network to respond not only to the local gradient, but also to the recent trends in the error surface, and ignore small features in the error surface. This term encourages movement in the same direction of successive steps. Without the value of $\beta$, a network may get stuck in a shallow local minimum. The use of $\beta$ might smooth out the oscillations and produce a stable trajectory [6]. As the $\beta$ coefficient increases, the oscillation in the output is reduced. By using $\beta$ in weight adaptation, a larger learning rate can be used while preserving the stability of the algorithm. $\beta$ also tends to accelerate convergence.

The advantages of using the $\beta$ term are summarized as follows:

1. Might smooth out oscillations occurs in learning,
2. Larger $\alpha$ value can be used if $\beta$ is added in weight adaptation,
3. Encourages movement in the same direction of successive steps.

3. Three Term BP Algorithm

The standard BP weight adaptation equation given by Equation (1) is modified by adding an extra term in order to increase the BP learning speed. The modification is done by adding a third term proposed by [1]. The third term, being the proportional term ($\gamma$) is proportional to:
\[ e(w(k)) = e_s, \]

where 
\[ e_s \text{ represents the difference between the output and the target at each iteration.} \]

Therefore, the new weight adaptation for 3-Term BP is defined as:

\[
\Delta W(k) = \alpha (-\nabla E(W(k))) + \beta \Delta W(k-1) + \gamma e_s(W(k)) \tag{2}
\]

where
\[ \alpha \text{ is proportional to the derivative of } E(W(k)), \]
\[ \beta \text{ is proportional to the previous value of the incremental change of the weights,} \]
\[ \gamma \text{ is proportional to } e_s. \]

From Equation (2), \( E(W(k)) \) can also be written as:

\[ E(W(k)) = \delta_k O_j. \tag{3} \]

For batch learning, \( e(W(k)) \) can be written as:

\[ e(W(k)) = [e_s e_s \ldots e_s]^\top, \]

where
\[ \text{vector } e \text{ is of appropriate dimension of } \tau, \text{ and} \]
\[ e_s = [T_j - O_j], \]

and 
\[ e_s \text{ represents the difference between the output and the target at each iteration.} \]

Some of the BP algorithm modifications require complex and costly calculations, which will only offset the faster rates of convergence that is obtained using the modified BP algorithms (refer Tab. II). As we can see from Equation (2), the 3-Term BP will maintain the simplicity of the standard BP algorithm. The 3-Term BP network had been tested on the XOR problem in [1], and it had significantly increased the convergence speed while maintaining the simplicity and efficiency of the standard BP. However, in our paper, we investigate further the analysis of the 3-term BP by presenting the results on three universal data with various sizes.

4. Methodology Development

A general framework of this study is shown in Fig. 1. The dataset used in the solving classification problem using the BP algorithm are Balloon dataset, Iris dataset and Cancer dataset. These data represent small, medium and large scale data based on their size. Balloon data is chosen to represent small scale data. The Balloon datasets were split with 12 for training data and 4 for testing data with 6 instances.
Learning Rate ($\alpha$) | Momentum Factor ($\beta$) | Proportional Factor ($\gamma$)
--- | --- | ---
Proportional to the derivative of $E(W(k))$ | Proportional to the previous value of the incremental change of the weights | Proportional to the difference between the output and the target at each iteration $e_s$
Added to increase convergence speed | Added to smooth out oscillation, increase convergence speed | Added to increase convergence speed, escape from local minima
Too large value will make network unstable | Enable the network to use larger $\alpha$ value | Does not effect the complexity of standard BP

Tab. II Characteristics of BP Learning Parameters.

Iris data represents medium scale data. This dataset contains data for three classes with 25 instances, where each class has 75 instances. Cancer data represents large scale dataset. The data are split into 2 category; 500 for training, and 100 for testing with 600 instances. These data are used to evaluate the performance of both standard and 3-Term BP algorithms in terms of classification accuracy and convergence speed.

4.1 Defining Neural Network Architecture

The network architecture consists of three layers; input layer, hidden layer and output layer. The number of nodes required in each layer differs from one dataset to another. Each defined input node represents the set of the problem that will be classified and the output node represents the classes that the input data will belong to after the classification has been done. The number of hidden nodes is defined as [8]:

$$\text{Number of hidden nodes} = \sqrt{m \ast n}$$

where

$m$ is the number of input nodes,
$n$ is the number of output nodes.

The Balloon dataset has 4 attributes with instances as color, size, act and age. Meanwhile the number of the output node for this dataset is only 1, since there is only one possible outcome for each combination of attributes; either the balloon is inflated (T) or not inflated (F). The network architecture for the Iris dataset comprises 4 input nodes, 3 output nodes and 3 hidden nodes. The Iris dataset has 4 attributes with each instance defined as petal width, petal length, sepal width and sepal length. The network architecture for the Cancer dataset comprises 9 input nodes, 1 output nodes and 3 hidden nodes.
4.2 Training BP Algorithm

The standard BP needs to be trained to minimize the measured error by adjusting the weights. In order to train the network, the initial weights and bias must be defined. Another parameter that is needed to be defined is the activation function. Our study used sigmoid function as an activation function. The maximum error is defined as it would be used as a stopping criterion. The training will be repeated until the network error is less than the maximum error. The same approach is used to train the 3-Term BP.

4.3 Implementing Proportional Factor ($\gamma$)

The proportional factor, $\gamma$ is implemented in the 3-Term BP alongside the other two terms $\alpha$ and $\beta$. The value of $\gamma$ is determined using the trial and error method. This study focuses on batch learning for Balloon, Iris and Cancer dataset. Batch learning will only do the training and testing phase. It holds no responsibility for performance during learning, unlike online learning. The BP algorithm is modified by adding an extra term in order to increase the BP learning speed. This term is proportional to $e(W(k))$ which represents the difference between the output and the target at each iteration. The error measure, $E$ used, is the Mean Square Error.
(MSE) and is defined as follows:

\[ E_p = \frac{1}{2} \sum_{j=1}^{N} (t_{kj} - o_{kj})^2, \]

where

\[ E_p \] is an error for the \( p^{th} \) presentation vector,
\( t_{kj} \) is the desired value from the output node \( (k) \) to the hidden node \( (j) \), and
\( o_{kj} \) is the network value from the output node \( (k) \) to the hidden node \( (j) \).

The weight changes are proportional to the derivative of \( E \). For example, the change in weights between output layer, \( k \) and hidden layer, \( j \) can be written as follows:

\[ \Delta W_{kj} = -\alpha \frac{\partial E}{\partial W_{kj}}, \]

where

\( \alpha \) is learning rate.

By using the chain rule, the above equation can be written as the following.

\[ \frac{\partial E}{\partial W_{kj}} = \frac{\partial E}{\partial net_k} \times \frac{\partial net_k}{\partial W_{kj}}. \] (4)

Let the error signal, \( \delta_k \) be,

\[ \delta_k = \frac{\partial E}{\partial net_k}. \] (5)

Since \( net_k = \sum_k W_{kj}O_j + \theta_k \) taking a partial derivation, will give,

\[ O_j = \frac{\partial net_k}{\partial W_{kj}}. \] (6)

By substituting Equation (6) and Equation (5) into Equation (4) gives

\[ \frac{\partial E}{\partial W_{kj}} = \delta_k \times O_j. \] (7)

From (5), we know that

\[ \delta_k = \frac{\partial E}{\partial net_k}. \]

This is obtained by applying a chain rule,

\[ \delta_k = \frac{\partial E}{\partial o_k} \times \frac{\partial o_k}{\partial net_k}. \] (8)

The partial derivative of error function, \( E = \frac{1}{2} \sum_k (t_{kj} - o_{kj})^2 \), can be written as this:

\[ \frac{\partial E}{\partial o_k} = - (t_k - o_k) \] (9)
The output of $k^{th}$ layer is given as this:

$$o_k = \frac{1}{1 + e^{-\text{net}_k}}.$$  

Therefore the partial derivative of $o_k$ is given as:

$$\frac{\partial o_k}{\partial \text{net}_k} = o_k (1 - o_k). \quad (10)$$

By substituting Equation (9) and Equation (10) into Equation (8), we get

$$\delta_k = - (t_k - o_k) o_k (1 - o_k). \quad (11)$$

By substituting Equation (11) into Equation (7), will give us,

$$\frac{\partial E}{\partial W_{kj}} = - (t_k - o_k) o_k (1 - o_k) \times O_j, \quad (12)$$

The weight adaptation between the output layer and the hidden layer can be written as this:

$$\Delta W_{kj} = -\alpha \frac{\partial E}{\partial W_{kj}} \quad (13)$$

By substituting Equation (12) into Equation (13),

$$\Delta W_{kj} (n) = -\alpha (- (t_k - o_k) o_k (1 - o_k) O_j), \quad (14)$$

By adding momentum term $\beta$ to Equation (14), the weight adaptation is given as this:

$$\Delta W_{kj} (n) = \alpha (t_k - o_k) o_k (1 - o_k) O_j + \beta \Delta W_{kj} (n - 1) \quad (15)$$

$\gamma$ Factor is added to equation (15) giving the weight adaptation as this:

$$\Delta W (k) = \alpha (-\nabla E (W (k))) + \beta \Delta W (k - 1) + \gamma e (W (k)) \quad (16)$$

where

$\alpha$ is proportional to the derivative of $E (W (k))$,

$\beta$ is proportional to the previous value of the incremental change of the weights,

$\gamma$ is proportional to $e_s$.

From equation (16), $E (W (k))$ can also be written as this:

$$E(W(k)) = \delta_k O_j \quad (17)$$

where

$\delta_k$ can be obtained from equation (8).
From Equation (16) for batch learning, $e(W(k))$ can be written as this:
\[
e(W(k)) = [e_s e_s \ldots e_s]^T
\]
where vector $e$ is of appropriate dimension of $\tau$, and
\[
e_s = [T_j - O_j],
\]
where $e_s$ represents the difference between the output and the target at each iteration.

The weights adaptation between hidden layer $j$, and input layer $i$, of the standard BP algorithm is similar as updating the weight between output layer $k$ and hidden layer $j$[9]. For comparison between the standard and the 3-Term BP, the same network structure is used and these include, a number of layers, a number of nodes for each layer, range of initial values, maximum error, learning rate and momentum value.

5. Results and Analysis

The performance evaluation of standard and Three Term BP are carried out based on its convergence rate and classification accuracy of the presented problem.

For standard BP, the experiments are divided into two testing sets which contain 9 trials for each dataset. For Test I, the same values of $\alpha$ and $\beta$ are used in the range of [0.1, 0.9]. While for Test II, the values of $\alpha$ and $\beta$ are increased and decreased, respectively with 0.1 as the initial values for $\alpha$ and 0.9 for $\beta$. For 3-Term BP, the experiments are divided into three testing datasets which also contains 9 trials for each dataset. For comparisons, the 3-Term BP used the same range of $\alpha$, $\beta$ and $\gamma$, i.e., [0.1, 0.9]. Test I used the same values for $\alpha$, $\beta$ and $\gamma$ in all 9 trials. Test II used the same increasing values for $\alpha$ and $\beta$ and a decreasing value for $\gamma$. $\alpha$ and $\beta$ values are started from 0.1, while the $\gamma$ value is initialized to 0.9. Test III used the increasing value for $\alpha$ and decreasing values for both $\beta$ and $\gamma$, where the $\alpha$ value is started from 0.1, while $\beta$ and $\gamma$ values are initialized to 0.9.

This analysis is carried out to investigate the efficiency of the 3-Term BP in solving classification problems. The comparisons are made on the results obtained from the experiments for both algorithms. Tab. III gives the percentage of correct classification for each dataset, and Fig. 2 depicts the classification rate for both networks. Fig. 3 shows the convergence rate of each dataset for both algorithms.

For Balloon dataset, the learning pattern indicates that the 3-Term BP converges faster than standard BP. The results show that the 3-Term BP generates less iteration, thus enhancing the learning speed. Both algorithms produced the same classification accuracy, which is 75% but in terms of convergence accuracy, the 3-Term BP gives a slightly better result with an error of 0.0426 compared to 0.0493 generated by the standard BP.
Tab. III Percentage of correct classification for standard BP and 3-Term BP.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Standard BP</th>
<th>3-Term BP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balloon</td>
<td>75%</td>
<td>75%</td>
</tr>
<tr>
<td>Iris</td>
<td>100%</td>
<td>96%</td>
</tr>
<tr>
<td>Cancer</td>
<td>50%</td>
<td>48%</td>
</tr>
</tbody>
</table>

For the Iris dataset, the learning pattern indicates that Standard BP converges faster than standard BP. The results show that the error convergence produced by standard BP is lower than Standard BP. In terms of the classification performance, standard BP gives a higher percentage of correct classification with a difference of 4% compared to the 3-Term BP. Thus, standard BP also performs better than 3-Term BP for Iris dataset which is a medium scale data, in terms of classification accuracy, and processing time.

For the Cancer dataset, the error produced by standard and the 3-Term BP started off with almost the same value but for 3-Term BP, the error value increased tremendously during iteration 782 making the characteristic convergence of the 3-Term BP ended with a high value or error of 8.5. Standard BP produced a higher percentage of correct classification compared to the 3-Term BP with a difference of 2%. Thus, in classifying the Cancer dataset which is a large scale data, the standard BP performs better than the 3-Term BP in terms of classification accuracy and processing time.

5.1 Stability Analysis

The stability analysis is carried out to investigate the stability of the 3-Term BP under various scale of dataset presented to the network. The main task of this analysis is to find the eigenvalues of $\lambda_1$ and $\lambda_2$ for Cancer and Iris datasets respectively. To prove this statement, the following derivations are given:
1. The conventional BP algorithm with two terms utilization has the following weight equation.

$$\Delta W(k) = -\alpha \nabla E(W(k)) + \beta \Delta W(k - 1), \quad (18)$$

where

$E$ is the average of mean square function (MSE),

$$\nabla E(W(k)) = \frac{1}{2k} \sum_{i=0}^{i=k} \sum_{s=0}^{s=i} (T(s) - O(s))^2,$$

and

$W$ is the network weight.

2. This study implements the third term $\gamma$ to increase the performance of standard BP. The term $\gamma$ is proportional to the $e(W(k)) = \frac{1}{k} \sum_{i=0}^{i=k} \sum_{s=0}^{s=i} (T(s) - O(s))$.

3. The new weight adaptation of the 3-Term BP is shown below:

$$\Delta W(k) = -\alpha \nabla E(W(k)) + \beta \Delta W(k - 1) + \gamma e(W(k)) \quad (19)$$

4. We want to analyze all local minima of the mean square error function that are only locally asymptotically stable points. Thus, equation (19) can be written as this:

$$W(k + 1) = W(k) - \alpha \nabla E(W(k))$$

$$+ \beta \Delta W(k - 1) + \gamma e(W(k)) \quad (20)$$

5. Local stability properties around an equilibrium point ($g_1, g_2$) can be examined by using a small signal analysis[10].

Let

$$g_1 = W(k) \text{ and } g_2 = W(k) - W(k-1),$$

then a state variable representation can be written as this:

$$g_1(k + 1) = g_1(k) - \alpha \nabla E(g_1(k))$$

$$+ \beta g_2(k) + \gamma e(g_1(k)) \quad (21)$$

Note that $g_2 = \Delta W(k - 1)$, then (21) can be rewritten as:

$$g_1(k + 1) - g_1(k) = -\alpha \nabla E(g_1(k)) + \beta g_2(k) + \gamma e(g_1(k))$$

$$\Delta g_1(k + 1) = -\alpha \nabla E(g_1(k)) + \beta g_2(k) + \gamma e(g_1(k))$$

$$\Delta W(k + 1) = -\alpha \nabla E(g_1(k)) + \beta g_2(k) + \gamma e(g_1(k))$$

11
6. From here we obtained another function:

\[ g_2(k + 1) = -\alpha \nabla E(g_1(k)) + \beta g_2(k) + \gamma e(g_1(k)) \]  
\hspace{1cm} (22)

Let

\[ A = \nabla E(g_1(k)) \text{ and } D = g_2(k). \]

Thus, equation (21) and Equation (22) can be represented into a linear equation as:

\[
\begin{bmatrix}
    g_1(k + 1) \\
    g_2(k + 1)
\end{bmatrix} =
\begin{bmatrix}
    1 - \alpha A + \gamma D & \beta \\
    -\alpha A + \gamma D & \beta
\end{bmatrix}
\begin{bmatrix}
    g_1(k) \\
    g_2(k)
\end{bmatrix}
\]  
\hspace{1cm} (23)

Equation (23) can be written in a more compact form as this:

\[ \phi(k + 1) = \Theta \phi(k) \]  
\hspace{1cm} (24)

7. It is well known that the discrete-time system in Equation (24) is asymptotically stable if \( \Theta \) has distinct eigenvalues, \( \lambda_i \) that satisfy this condition \cite{10}, \(|\lambda_i| < 1\).

Let \( \Theta \) be the matrix 2x2 that correspond to \( \Theta = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \), then the eigenvalues can be obtained from

\[ \lambda_i = \frac{1}{2} (a + d \pm \sqrt{4bc + (a - d)^2}) \]  
\hspace{1cm} (25)

By applying Equation (23) into Equation (25), we will get,

\[ \lambda_i = \frac{1}{2} (T \pm \sqrt{U + V}), \]  
\hspace{1cm} (26)

where

\[ T = 1 - \alpha A + \gamma D + \beta \]
\[ U = 4\beta(\gamma D - \alpha A) \]
\[ V = 1 - \alpha A + \gamma D - \beta \]

Eigenvalues for medium and large scale data are calculated for system’s generated errors. The results indicate that the system is stable when represented with small scale data, but not in a stable state when the system is represented with medium or large data, i.e., \(|\lambda| > 1\). This proves that the 3-Term BP only outperforms standard BP for small scale dataset, but not for medium and large scale dataset (Tab. IV).
**6. Conclusion**

The objectives of this study are to investigate the efficiency of the 3-Term BP and its performance ability in classification problems. The results are compared with standard BP. Tab. V shows the performance analysis of three datasets that have been used in this study using the standard BP and the 3-Term BP.

In this paper, the implementation of the $\gamma$ factor in the 3-Term BP as the third term only enhances the performance in small scale data, but not for medium and large scale data. For future works, there are several suggestions that can be done to further improve this study. An optimization method can be used to fine tune various network parameters such as initial weight, maximum error, number of hidden layer and hidden nodes to obtain optimal value such as Genetic Algorithm (GA) or Particle Swarm Optimization (PSO). Huge input data that represent various scale dataset can also be considered as well as other factors in exploring the behavior of the 3-Term BP such as a local minima problem and a type of the error function in order to obtain better results.

<table>
<thead>
<tr>
<th>Analysis Criteria</th>
<th>Balloon</th>
<th>Iris</th>
<th>Cancer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification Performance</td>
<td>3-Term BP better</td>
<td>Std BP better</td>
<td>Std BP better</td>
</tr>
<tr>
<td>Processing Time</td>
<td>3-Term BP faster</td>
<td>Std BP faster</td>
<td>Std BP faster</td>
</tr>
<tr>
<td>Error Convergence</td>
<td>3-Term BP lower</td>
<td>Std BP lower</td>
<td>Std BP lower</td>
</tr>
</tbody>
</table>


