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

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

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 3

Weight Changes for Learning Mechanisms in Two-Term Back-Propagation Network

Siti Mariyam Shamsuddin,
Ashraf Osman Ibrahim and Citra Ramadhena

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/51776

1. Introduction

The assignment of value to the weight commonly brings the major impact towards the learning behaviour of the network. If the algorithm successfully computes the correct value of the weight, it can converge faster to the solution; otherwise, the convergence might be slower or it might cause divergence. To prevent this problem occurring, the step of gradient descent is controlled by a parameter called the learning rate. This parameter will determine the length of step taken by the gradient to move along the error surface. Moreover, to avoid the oscillation problem that might happen around the steep valley, the fraction of last weight update is added to the current weight update and the magnitude is adjusted by a parameter called momentum. The inclusion of these parameters aims to produce a correct value of weight update which later will be used to update the new weight. The correct value of weight update can be seen in two aspects: sign and magnitude. If both aspects are properly chosen and assigned to the weight, the learning process can be optimized and the solution is not hard to reach. Owing to the usefulness of two-termBP and the adaptive learning method in learning the network, this study is proposing the weights sign changes with respect to gradient descent in BP networks, with and without the adaptive learning method.

2. Related work

Gradient descent technique is expected to bring the network closer to the minimum error without taking for granted the convergence rate of the network. It is meant to generate the

slope that moves downwardsalong the error surface to search for the minimum point. During its movement, the points passed by the slope throughout the iterations affect the magnitude of the value of weight update and its direction. Later, the updated weight is used for training the network at each epoch until the predefined iteration is achieved or the minimum error has been reached. Despite the general success of BP in learning, several major deficiencies still need to be solved. The most notable deficiencies, according to reference [1], are the existence of temporary local minima due to the saturation behaviour of activation function. The slow rates of convergence are due to the existence of local minima and the convergence rate is relatively slow for a network with more than one hidden layer. These drawbacks are also acknowledged by several scholars [2-5].

Error function plays a vital role in the learning process of two-termBP algorithm. A side from calculating the actual error from the training, it assists the algorithm in reaching the minimum point where the solution converges by calculating its gradient and back propagation to the network for weight adjustment and error minimization. Hence, the problem of being trapped in local minima can be avoided and the desired solution can be achieved.

The movement of the gradient on the error surface may vary in term of its direction and magnitude. The sign of the gradient indicates the direction it moves and the magnitude of the gradient indicates the step size taken by the gradient to move on the error surface. This temporal behaviour of the gradient provides insight about conditions on the error surface. This information will then be used to perform a proper adjustment of the weight, which is carried out by implementing a weight adjustment method. Once the weight is properly adjusted, the learning process takes only a short time to converge to the solution. Hence, the problem faced by two-termBP is solved. The term "proper adjustment of weight" here refers to the proper assignment of magnitude and sign to the weight, since both of these factors affect the internal learning process of the network.

Aside from the gradient, there are some factors that play an important role in the assignment of proper change to the weight specifically in term of its sign. These factors are the learning parameters such as learning rate and momentum. Literally, learning rate and momentum parameters hold an important role in the two-term BP training process. Respectively, they control the step size taken by the gradient along the error surface and speed up the learning process. In a conventional BP algorithm, the initial value of both parameters is very critical since it will be retained throughout all the learning iterations. The assignment of fixed value to both parameters is not always a good idea,bearing in mind that the error surface is not always flat or never flat. Thus, the step size taken by the gradient cannot be similar over time. It needs to take into account the characteristics of the error surface and the direction of movement. This is a very important condition to be taken into consideration to generate the proper value and direction of the weight. If this can be achieved, the network can reach the minimum in a shorter time and the desired output is obtained.

Setting a larger value for the learning rate may assist the network to converge faster. However, owing to the larger step taken by the gradient, the oscillation problem may occur and cause divergence or in some cases, we might overshoot the minimum. On the other hand, if the smaller value is assigned to the learning rate, the gradient will move in the correct direc-

http://dx.doi.org/10.5772/51776

tion and gradually reach the minimum point. However, the convergence rate is compromised owing to the smaller steps taken by the gradient. On the other hand, the momentum is used to overcome the oscillation problem. It pushes the gradient to move up the steep valley in order to escape the oscillation problem, otherwise the gradient will bounce from one side of the surface to another. Under this condition, the direction of gradient changes rapidly and may cause divergence. As a result, the computed weight update value and direction will be incorrect, which affects the learning process. It is obviously seen that the use of a fixed parameter value is not efficient. The obvious way to solve this problem is to implement an adaptive learning method to produce the dynamic value of learning parameters.

In addition, the fact that the two-term BP algorithm uses the uniform value of learning rate may lead to the problem of overshooting minima and slow movement on the shallow surface. This phenomenon may cause the algorithm to diverge or converge very slowly to the solution owing to the different step size taken by each slope to move in a different direction. In [6] has proposed a solution to these matters, called the Delta-Bar-Delta (DBD) algorithm. The method proposed by the author focuses on the setting of a learning rate value for each weight connection. Thus, each connection will have its own learning rate. However, this method still suffers from certain drawbacks. The first drawback is that the method is not efficient to be used together with the momentum since sometimes it causes divergence. The second drawback is the assignment of the increment parameter which causesa drastic increment on the learning rate so that the exponential decrement does not bring a significant impact to overcome a wild jump. For these reasons, [7] proposed an improved DBD algorithm called the Extended Delta-Bar-Delta (EDBD) algorithm. EDBD implements a similar notion of DBD and adds some modifications to it to alleviate the drawbacks faced by DBD, and demonstrates a satisfactory learning performance. Unlike DBD, EDBD provides a way to adjust both learning rate and momentum for individual weight connection, and its learning performance is thus superior to DBD. EDBD is one of many adaptive learning methods proposed to improve the performance of standard BP. The author has proven that the EDBD algorithm outperforms the DBD algorithm. The satisfactory performance tells us that the algorithm performssuccessfully and well in generating proper weight with the inclusion of momentum.

[8] has proposed the batch gradient descent method momentum, by combining the momentum with the batch gradient descent algorithm. Any sample in the network cannot have an immediate effect, however; it has to wait until all the input samples are in attendance. If that happens, then we accumulate the sum of all errors, and finally focus on the right to modify the weights to enhance the convergence rate accordingto the totalerror. The advantages of this method are faster speed, fewer iterations and smoother convergence. On the other hand, [9] has presented a new learning algorithm for a feed-forward neural network based on the two-termBP method using an adaptive learning rate. The adaptation is based on the error criteria where error is measured in the validation set instead of the training set to dynamically adjust the global learning rate. The proposed algorithm consists of two phases. In the first phase, the learning rate is adjusted after each iterationso that the minimum error is quickly attained. In the second phase, the search algorithm is refined by repeatedly reverting to previous weight configurations and decreasing the global learning rate. The experi-

mental result shows that the proposed method quickly converges and outperforms two-term BP in terms of generalization when the size of the training set is reduced. [10] has improved the convergence rates of the two-term BP model with some modifications in learning strategies. The experiment results show that the modified BP improved much better compared with standard BP.

Meanwhile, in [11] proposed a differential adaptive learning rate method for BP to speed up the learning rate. The proposed method employs the large learning rate at the beginning of training and gradually decreases the value of learning rate using differential adaptive method. The comparison made between this method and other methods, such as two-term BP, Nguyen-Widrow Weight Initialization and Optical BP shows that the proposed method outperforms the competing method in terms of learning speed.

[5] proposed a new BP algorithm with adaptive momentum for feed-forward training. Based on the information of current descent direction and last weight increment, the momentum coefficient is adjusted iteratively. Moreover, while maintaining the stability of the network, the range for the learning rate is widened after the inclusion of the adaptable momentum. The simulation results show that the proposed method is superior to the conventional BP method where fast convergence is achieved and the oscillation is smoothed.

[12] presented an improved training algorithm of BP with a self-adaptive learning rate. The function relationship between the total quadratic training error change and the connection weight and bias change is acquired based on the Taylor formula. By combining it with weight and bias change in a batch BP algorithm, the equations to calculate a self-adaptive learning rate are obtained. The learning rate will be adaptively adjusted based on the average quadratic error and the error curve gradient. Moreover, the value of the self-adaptive learning rate depends on neural network topology, training samples, average quadratic error and gradient but not artificial selection. The result of the experiment shows the effectiveness of the proposed training algorithm.

In [13] a fast BP learning method is proposed using optimization of the learning rate for pulsed neural networks (PNN). The proposed method optimized the learning rate so as to speed up learning in every learning cycle, during connection weight learning and attenuation rate learning for the purpose of accelerating BP learningina PNN.The authors devised an error BP learning method using optimization of the learning rate. The results showed that the average number of learning cycles required in all of the problems was reduced by optimization of the learning rate during connection weight learning, indicating the validity of the proposed method.

In [14], the two-term BP is improvedso that it can overcome the problems of slow learning and is easy to trap into the minimum by adopting an adaptive algorithm.The method divides the whole training process into many learning phases. The effects will indicate the direction of the network globally. Different ranges of effect values correspond to different learning models. The next learning phase will adjust the learning model based on the evaluation effects according to the previous learning phase.

http://dx.doi.org/10.5772/51776

We can infer from previous literature that the evolution of the improvement of BP learning for more than 30 years still points towards the openness of contribution in enhancing the BP algorithm in training and learning the network especially in terms of weight adjustments. The modification of the weight adjustment aims to update the weight with the correct value to obtain a better convergence rate and minimum error. This can be seen from various studies that significantly control the proper sign and magnitude of the weight.

3. Two-TermBack-Propagation (BP) Network

The architecture of two-term BP is deliberately built in such away that it resembles the structure of neuron. It contains several layers where each layer interacts with the upper layer connected to it by connection link. Connection link is specifically connecting the nodes with in the layers with the nodes in the adjacent layer that builds a highly inter connected network. The bottom-most layer, called the input layer, will accept and process the input and pass the output to the next adjacent layer, called the hidden layer. The general architecture of ANN is depicted in Figure1 [15].

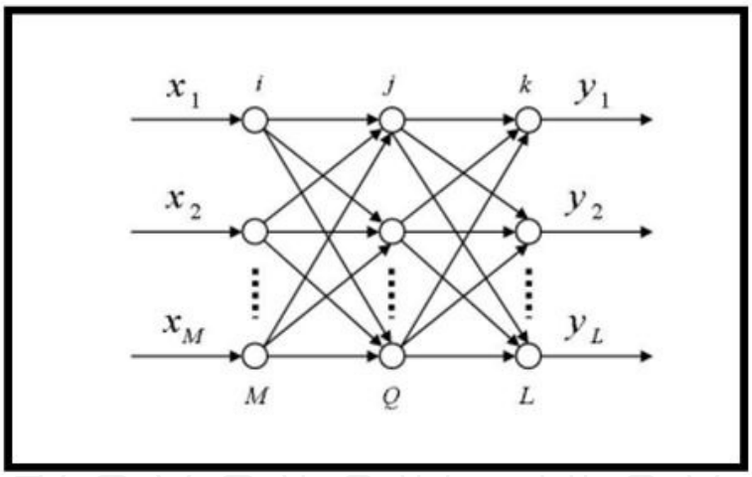

Figure 1. ANN architecture

where,

i is the input layer.

j is the hidden layer and

k is the output layer.

The input layer has M neurons and input vector $X = [\, x_1 , x_2 ,\ldots, x_M \,]$ and the output layer has L neurons and has output vector $Y=[\, y_1 , y_2 ,\ldots, y_L \,]$ while the hidden layer has Q neurons.

The output received from the input layer will be processed and computed mathematically in the hidden layer and the output will be passed to the output layer. In addition, BP can have more than one hidden layer but it creates complexity in training the network. One reason for this complexity is the existence of local minima compared with the one with one hidden layer. The learning depends greatly on the initial weight choice to lead to convergence.

Nodes in BP can be thought of a sun its that process in put to produce output. The output produced by the node is affected largely by the weight associated with each link. In this process, each input will be multiplied with weight associated with connection link connected to the node and added with bias. Weight is used to determine the strength of the output to be closer to the desired output. The greater the weight, the greater the chance of the output being closer to the desired output. The relationship between the weight, connection link and the layers can be shown in Figure 2 in reference [16].

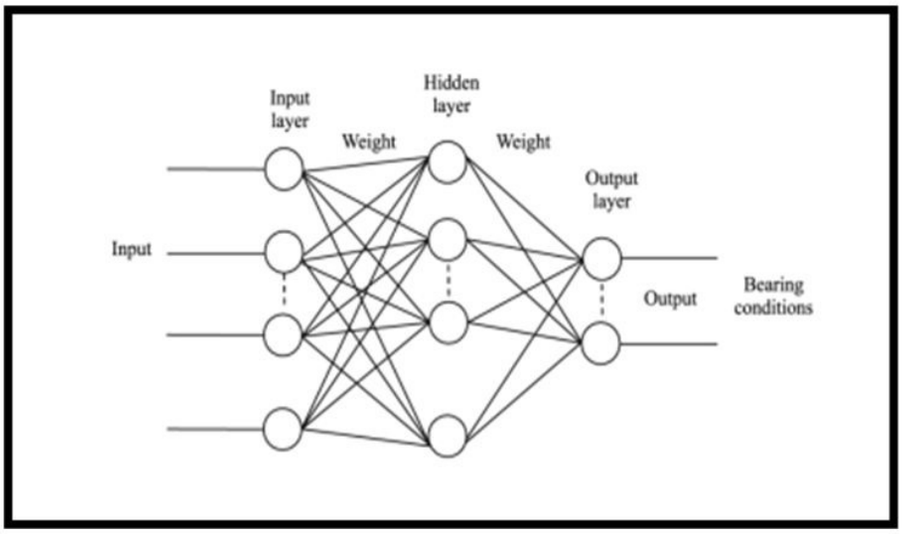

Figure 2. Connection links, weights and layers.

Once the output arrives at the hidden layer, it will be summed up to create a net.This is called linear combination. The net is fed to activation function and the output will be passed to the output layer. To ensure the learning takes place continuously, in the sense that the derivative of error function can keep moving down hillon the error surface in searching for the minimum, the activation function needs to be a continuous differentiable function. The most commonly used activation function is the sigmoid function, which limits the output between 0 and 1.

$$net_j = \sum_i W_{ij} O_i + \theta_i \tag{1}$$

$$O_j = \frac{1}{1 + e^{-net_j}} \tag{2}$$

http://dx.doi.org/10.5772/51776

where,

net_j is the summation of the weighted input added with bias,

W_{ij} is the weight associated at the connection link between nodes in the input layer*i*and nodes in

hidden layer *j*,

O_i is the input at the nodes in input layer*i*,

θ_i is the bias associated at each connection link between input layer*i*andhidden layer*j*,

O_j is the output of activation functionat hidden layer *j*

Other activation functions that are commonly used are the logarithmic,tangent, hyperbolic tangent functions and many more.

The output generated by activation functionis forwarded to the output layer. Similar to input and hidden layers, the connection link that connects the hidden and output layers is associated with weight. Activated output received from the hidden layer is multiplied by weight. Depending on the application, the number of nodes in the output layer may vary. In a classification problem, the output layer only consists of one node to produce the result of either yes or no or abinary numbers. All the weighted outputs are added together and this value will be fed to the activation function togener ate the final output. Mean Square Error is used as an error function to calculate the error at each iteration using the target output and the final calculated output of the learning at each iteration. If the error is still larger than the predefined acceptable error value, the training process continues to the next iteration.

$$net_k = \sum_j W_{jk} O_j + \theta_k \tag{3}$$

$$O_k = \frac{1}{1 + e^{-net_k}} \tag{4}$$

$$E = \frac{1}{2} \sum_k \left(t_k - o_k \right)^2 \tag{5}$$

where,

net_k is the summation of weighted output at the output layer *k*,

O_j isthe output at nodes in hidden layer *j*,

W_{jk} is the weight associated to connection link between the hidden layer *j* and

the output layer *k*,

E is the error function of the network (Mean Square Error),

t_k is the target output at output layer k,

θ_k is the bias associated to each connection link between the hidden layer

j and the output layer k,

O_k is the final output at the output layer.

A large value of error obtained at the end of each iteration denotes the deviation of learning where the desired output has not been achieved. To solve this problem, the derivative of error function with respect to weight is computedand back-propagated to the layers to compute the new weight value at each connection link. This algorithm is known as the delta rule, which employs the gradient descent method. The new weight is expected to be a correct weight that can produce the correct output. For weight associated to each connection link between output layer k to hidden layer j, the weight incremental value is computed using a weight adjustment equation as follows:

$$\Delta W_{kj}(t) = -\eta \frac{\partial E}{\partial W_{kj}} + \beta \Delta W_{kj}(t-1) \tag{6}$$

where,

$\Delta W_{kj}(t)$ is the weight incremental value at tth iteration

$\eta$is the learning rate parameter

$-\frac{\partial E}{\partial W_{kj}}$ is the negative derivative of error function with respect to weight

β is the momentum parameter

$\Delta W_{kj}(t-1)$ is the previous weight incremental value at (t-1)th iteration

By applying the chain rule,we can simplify the negative derivative of error function with respect to weight as follows:

$$\frac{\partial E}{\partial W_{kj}} = \frac{\partial E}{\partial net_k} \bullet \frac{\partial net_k}{\partial W_{kj}} \tag{7}$$

Substituting Equation (3) into Equation (6), we get,

$$\frac{\partial E}{\partial W_{kj}} = \frac{\partial E}{\partial net_k} \bullet O_j \tag{8}$$

$$\frac{\partial E}{\partial net_k} = \delta_k \tag{9}$$

Thus, by substituting Equation (9) into Equation (8), we get,

http://dx.doi.org/10.5772/51776

$$\frac{\partial E}{\partial W_{kj}} = \delta_k \bullet O_j \tag{10}$$

Simplifying Equation (9) to be as follows:

$$\delta_k = \frac{\partial E}{\partial net_k} = \frac{\partial E}{\partial O_k} \bullet \frac{\partial O_k}{\partial net_k} \tag{11}$$

$$\frac{\partial O_k}{\partial net_k} = O_k(O_k - 1) \tag{12}$$

Substituting Equation (12) into Equation (11) yields,

$$\frac{\partial E}{\partial net_k} = \frac{\partial E}{\partial O_k} \bullet O_k(O_k - 1) \tag{13}$$

$$\frac{\partial E}{\partial O_k} = -(t_k - O_k) \tag{14}$$

Substituting Equation (14) in to Equation (13), we get error signal at output layer

$$\delta_k = \frac{\partial E}{\partial net_k} = -(t_k - O_k) \bullet O_k(O_k - 1) \tag{15}$$

Thus, by substituting Equation (15) into Equation (6), we get the weight adjustment equation forweight associated to each connection link between output layer *k* to hidden layer *j* with simplified negative derivative error function with respect to weight,

$$\Delta W_{kj}(t) = \eta(t_k - O_k) \bullet O_k(O_k - 1) \bullet O_j + \beta \Delta W_{kj}(t - 1) \tag{16}$$

On the other side, the error signal is back-propagated to bring impact to the weight between input layer *i* and hidden layer *j*. The error signal at hidden layer*j*can be written as follows:

$$\delta_j = \left(\sum_k \delta_k W_{kj}\right) O_j(1 - O_j) \tag{17}$$

Based on Equation(10) and the substitution of Equation(17) in to Equation(6),the weight adjustment equation for the weights associated to each connection link between input layer *I* and hidden layer *j* is as follows:

$$\Delta W_{ji}(t) = \eta\left(\sum_k \delta_k W_{kj}\right) O_j(1 - O_j) \bullet O_i + \beta \Delta W_{ji}(t - 1) \tag{18}$$

Where,

$\Delta W_{ji}(t)$ is the weight incremental value a*t* *t*th iteration,

O_i is the input at nodes in input layer *i*.

The values obtained from Equation (16) and Equation (18) are used to update the value of weights at each connection link. Let t refer to *t*th iteration of training; the new weight value at (*t+1*)th iteration associated to each connection link between output layer to hidden layer *j* is calculated as follows:

$$W_{kj}(t+1) = \Delta W_{kj}(t) + W_{kj}(t) \tag{19}$$

Where,

$W_{kj}(t+1)$ is the new value for weight associated to each connectionlinkbetween output layer *k* and hidden layer *j*,

$W_{kj}(t)$ is the current value of weight associated to each connection linkbetween output layer *k* and hidden layer *j* at *t* th iteration

Meanwhile, the new weight value at (*t*-1)th iteration for the weight associated at each connection link between hidden layer *j* and the input layer *i* can be written as follows:

$$W_{ji}(t+1) = \Delta W_{ji}(t) + W_{ji}(t) \tag{20}$$

Where,

$W_{ji}(t+1)$ is the new value for the weight associated to each connection linkbetween hidden layer *j* and input layer *i*,

$W_{ji}(t)$ is the current value of weight associated to each connection link between hidden layer *j* and input layer *i.*

The gradient of error function is expected to move down the error surface and reach the minima point where the global minimum resides. Owing to the temporal behaviour of gradient descent and the shape of the error surface, the step taken to move down the error surface may lead to the divergence of the training. Many reasons can cause this problem but one of the misovershooting the local minima where the desired output lies. This may happen when the step taken by the gradient is large. However, a large step can lead the network to converge faster but when it moves down along the narrow and steep valley, the algorithm might go in the wrong direction and bounce from one side across to the other side.In contrast, a small step can direct the algorithm to the correct direction but the convergencerate is compromised. The learning time becomes slower since more instances of training are needed to achieve minimum error. Thus, the difficulty of this algorithm lies in controlling the step and direction of the gradient along the error surface. For this reason a parameter called the learning rate is used in weight adjustment computation. The choice of learning rate value is application-dependent and most cases are based on experiments. Once the cor-

http://dx.doi.org/10.5772/51776

rect learning rate is obtained, the gradient movement can produce the correct new weight value to produce the correct output.

Owing to the problem of oscillation in the narrow valley, another parameter is needed to keep the gradient moving in the correct direction so that the algorithm will not suffer from wide oscillation. This parameter is called momentum. Momentum brings the impact of previous weight change to the current weight change by which the gradient will move uphill escaping the oscillation along the valley. The incorporation of two parameters in the weight adjustment calculation produces a great impact on the convergence of the algorithm and problem of local minima if they are tuned to the correct value.

4. Weight Sign and Adaptive Methods

The previous sections have discussed the role of parameters in producing the increment value of weight through the implementation of a weight adjustment equation. As discussed before, learning rate and momentum coefficient are the most commonly used parameters in two-term BP. The use of a constant value of parameter is not always a good idea. In the case of learning rate, setting up a smaller value to learning rate may decelerate the convergence speed even though it can guarantee that the gradient will move in the correct direction. On the contrary, setting up a larger value to learning rate may fasten the convergence speed but is prone to an oscillation problem that may lead to divergence. On the other hand, the momentum parameter is introduced to stabilize the movement of gradient descentin the steepest valley by overcoming the oscillation problem. In [15] stated that assigning too small a value to the momentum factor may decelerate the convergence speed and the stability of the network is compromised, while too large a value for the momentum factor results in the algorithm giving excessive emphasis to the previous derivatives that weaken the gradient descent of BP. Hence, the author suggested the use of a dynamic adjustment method for momentum. Like the momentum parameter, the value of the learning rate also needs to be adjusted at each iteration to avoid the problem produced by having a constant value throughout all iterations.

The adaptive parameters (learning rate and momentum) used in this study are implemented to assist the network in controlling the movement of gradient descent on the error surface which primarily aims to attain the correct value of the weight.

The correct increment value of weight will be used later to update the new value of weight. This method will be implemented to two-term BP algorithmwith MSE. The adaptive method assists in generating the correct sign value for the weight, which is the primary concern of this study.

The choice of the adaptive method focuses on the learning characteristic of the algorithm used in this study, which is batch learning. In [17] gave a brief definition of online learning and the difference with batch learning. The author defined online learningas a scheme for updating weight that updates weight after every input-output case, while batch learning ac-

cumulates error signals over all input-output cases before updatingweight.In otherwords, online learningupdates weight after the presentation of each input and target data. The batch learning method reflects the true gradient descent where, asstated in reference [18], each weight update tries to minimize the error.The author also stated that the summed gradient information for the whole pattern set provides reliable information regarding the shape of the whole error function.

With the task of pointing out the temporal behaviour of gradient of error functionanditsrelationtothechangeofweightsign,theadaptivelearning method used in this study is adopted from the paper written by reference [7]entitled "Back-Propagation Heuristics: A Study of the Extended Delta-Bar-Delta Algorithm". The author proposed an improvement of the DBD algorithm proposed in reference [6], called the Extended Delta-Bar-Delta (EDBD) algorithm, where the improved method provide sa way of updating the value of momentum for each weight connection a teach iteration. Since EDBD is anextension of DBD, it implements a similar notion to DBD. It exploits the information of the sign of past and current gradients. The sign information of past and current gradients becomes the condition for learning rate and momentum adaptation. Moreover,the improved algorithm also providesace iling to prevent the value of learning rate and momentum becoming too large. The detail edequations of the method are described below:

$$\Delta w_{ji}(t) = -\eta_{ji}(t)\frac{\partial E(t)}{\partial w_{ji}} + \beta_{ji}(t)\Delta w_{ji}(t-1) \tag{21}$$

Where,

η_{ji} is the learning rate between *i*th input layer and *j*th hidden layer a *t* *t*thiteration

$\beta_{ji}(t)$ is the momentum between *i*th input layer and*j*th hidden layer at *t* th iteration

The updated value for learning rate and momentum can be written as follows.

$$\Delta\eta_{ji}(t) = \begin{cases} k_l \exp\left(-y_l \left|\bar{\delta}_{ji}(t)\right|\right) & if\ \bar{\delta}_{ji}(t-1)\delta_{ji}(t) > 0 \\ -\varnothing_l \eta_{ji}(t) & if\ \bar{\delta}_{ji}(t-1)\delta_{ji}(t) < 0 \\ 0 & otherwise \end{cases} \tag{22}$$

$$\Delta\beta_{ji}(t) = \begin{cases} k_m \exp\left(-y_m \left|\bar{\delta}_{ji}(t)\right|\right) & if\ \bar{\delta}_{ji}(t-1)\delta_{ji}(t) > 0 \\ -\varnothing_m \beta_{ji}(t) & if\ \bar{\delta}_{ji}(t-1)\delta_{ji}(t) < 0 \\ 0 & otherwise \end{cases} \tag{23}$$

$$\bar{\delta}_{ij}(t) = (1-\theta)\delta_{ij}(t) + \theta\bar{\delta}_{ij}(t-1) \tag{24}$$

http://dx.doi.org/10.5772/51776

$$\eta_{ji}(t+1) = MIN[\eta_{max},\ \eta_{ji}(t) + \Delta\eta_{ji}(t)] \tag{25}$$

$$\beta_{ji}(t+1) = MIN[\beta_{max},\ \beta_{ji}(t) + \Delta\beta_{ji}(t)] \tag{26}$$

Where,

k_l, y_l, $\varnothing_l$ are parameters for learning rate adaptive equation,

k_m, y_m, $\varnothing_m$ are parameters for momentum adaptive equation,

$\theta$is the weighting on the exponential average of the past derivatives,

$\bar{\delta}_{ij}$ is the exponentially decaying trace of gradient values,

δ_{ij} is gradient value between i th input layer andj th hidden layer

at ^{t} th iteration,

β_{max} is the maximum value of momentum,

η_{max} is the maximum value of learning rate.

The algorithm calculates the exponential average of past derivatives to obtain information about the recent history of the direction in which the error is decreasing up to iteration *t*.This information together with the current gradient is used to adjust the parameters'value based on their sign. When the current and past derivatives possess the same sign, it shows that the gradient is moving in the same direction. One can assume that in this situation, the gradient is moving in the flat area at which the minimum lies ahead. In contrast, when the current and past derivatives possess an opposite sign, it shows that the gradient is moving ina different direction.

One can assume that in this situation,the gradient has jumped over the minimuma nd weight needs to be decreased to solve this.

The increment of learning rate value is made proportional to exponentially decaying trace so that the learning rate will increase significantly at a flat region and decrease at a steep slope. To prevent the unbounded increment of parameters, the maximum value for learning rate and momentum are set to act as a ceiling to both parameters.

Owing to the excellent idea and performance of the algorithmas has been proven in reference [7], this method is proposed to assist the network in producing proper weight sign change and achieving the purpose of this study.

5. Weights Performance in Terms of Sign Change and Gradient Descent

Mathematically, the adaptivemethodandtheBPalgorithm, specifically the weight adjustment method described in Equations (19) and (20) used in this study, will assist the algorithm in

producing the proper weight. Basically, the adaptive methods are the transformation of the author's idea of an optimization conceptinto a mathematical concept. The measurement of the success of the method and the algorithm as a whole can be done in many ways. Some of them are carried out by analysing the convergence rate, the accuracy of the result, the error value it produces and the change of the weight sign as a response to the temporal behaviour of the gradient, etc. Hence, the role of the adaptive method that is constructed by using a mathematical concept to improve the weight adjustment computation in order to yield the proper weights will be implemented and examined in this study to check the efficiency and the learning behaviour of both algorithms. The efficiency can be drawn from the criteria of the measurement of success described earlier.

A simplicitly depicted from the algorithm, the process of generating the proper weight stems from calculating it sup date value. This process is affected by various variables starting from the parameters up to the controlled variable such as gradient, previous weight increment value and error. They all play a great part in affecting the sign of the weight produced, especially the partial derivative of error function with respectto weight (gradient). In reference [15] briefly described the relationship between error curvature, gradient and weight. The author mentioned that when the error curve enters the flat region, the change of derivative and error curve are smaller and as a result, the change of the weight will not be optimized. Moreover, when it enters the high curvature region, the derivative change is large especially if the minimum point exists at this region, and the adjustment of weight value is large which sometimes overshoots the minima. This problem can be alleviated by adjusting the step size of the gradient and this can be done by adjusting the learning rate. The momentum coefficient can be used to control the oscillation problem and its implementation along with the proportional factor can speed up the convergence. In addition, in [15] also gave the proper condition for the rate of weight and the temporal behaviour of the gradient. The author wrote that if the derivative has the same symbol as the previous one, then the sum of the weight is in creased, which makes the weight increment value larger and yields the increment of weight rate. On the contrary, if the derivative has the opposite sign to the previous one, the sum of the weight is decreased to stabilize the network. In [19] also emphasized the causes of the slow convergence which in volve the magnitude and the direction component of the gradient vector. The author stated that when the error surface is fairly flat along the weight dimension, the magnitude of derivative of weight is small yields small adjustment value of weight and many steps are required to reduce the error. Meanwhile, when the error surface is highly curved along the weight dimension, the derivative of weight is large in magnitude yields a large value of weight which may over shoot the minimum. The author also briefly discussed the performance of BP with momentum. The author stated that when the consecutive derivatives of a weight possess the same sign, the exponentially weighted sum grows large in magnitude and the weight is adjusted by a large value.

On the contrary, when the signs of the consecutive derivatives of the weight are opposite, the weighted sum is small in magnitude and the weight is adjusted by a small amount. Moreover, the author raised the implementation of local adaptive methods such as Delta-Bar-Delta which is originally proposed in reference [6]. From the descriptiongivenin [6], the

http://dx.doi.org/10.5772/51776

learning behaviour of the network can be justified as follows: The consecutive weight increment value that possesses the opposite sign indicates the oscillation of weight value which requires the learning rate to be reduced. Similarly, the consecutive weight increment value that possesses the same sign requires the incremental of the value of the learning rate. This information will be used in studying the weight sign change of both algorithms.

6. Experimental Result and Analysis

This section discusses the result of the experiment and its analysis. The detailed discussion of the experiment process covers its initial phase until the analysis is summarized.

The point of learning occurs when the difference between the calculate do utput and the desired output exists,otherwise there is no point of learning to take place. When the difference does exist, the error signal is propagated back into the network to be minimized. The network will then adjust itself to compensate the lost during the training to learn better. This procedure is carried out by calculating the gradient of error which mainly aimed to adjust the value of the weight to be used for the next feed-forward training procedure.

By looking at the feed-forward computation in Equation (1) through Equation (5), the next data train is fed again into the network and multiplied by the new weight value. The similar feed-forward and backward procedures are performed until the minimum value of error is achieved. A sa result,thetraining may take fewer or more iterations depending on the proper adjustment of weight. There is no doubt that weight plays important role intraining. Its role lies in determining the strength of the incoming signal (input) in the learning process.This weighted signal will be accumulated and forwarded to the next adjacent upper layer.

Based on the influence of weight, this signal can bring a bigger or smaller influence to the learning. When the weight is negative, the weight connection inhibits the input signal, and thus it does not bring significant influence to the learning and output. As are sult, the other nodes with positive weight will dominate the learning and the output. On the other hand, when the weight is positive, the weight exhibits the input signal to bring significant impact to the learning and the output and the respective node makes a contribution to the learning and output. If the assignment results in large error, the corresponding weight needs to be adjusted to reduce the error. The adjustment comprises magnitude and sign change.By properly adjusting the weight, the algorithm can converge to the solution faster.

To have the clear idea of the impact of the weight sign in the learning,the assumption is used: Let all value of weights be negative and using the equation written below, we obtain the negative value of *net*.

$$net = \sum_i W_i O + \theta \tag{27}$$

$$O = \frac{1}{1 + e^{-net}} \tag{28}$$

Feeding *net* into the sigmoid activation function Equation (28), we obtain the value of *O* close to 0. On the other hand, Letall value of weights be negative and using the equation below, we obtain the positive value of*net*.

Feeding *net* into the sigmoid activation function at equation (28), we obtain the value of *O* close to1. From the assumption above, we can infer that by adjusting the weight with the proper value and sign, the network can learn better and faster. Mathematically, the adjustment of weight is carried out by using the weight update method in and the current weight value as written below.

$$\Delta W(t) = \eta(t) \nabla E(W(t)) + \beta(t) \Delta W(t - 1) \tag{29}$$

$$W(t + 1) = \Delta W(t) + W(t) \tag{30}$$

It can be seen from Equation (29) and (30) that the rear evarious factors which influence the calculation of the weight update value. The most notable factor is gradient descent. Each gradient with respect to each weight in the network will be calculated and the weight update value is computed to update the corresponding weight. The negative sign is assigned to a gradient to force it to move downhill along the error surface in the weight space. This is meant to find the minimum point of the error surface where the set of optimal weight resides. With this optimal value, the goal of learning can be achieved. Another factor is the previous weight update value.

According to reference [20], the role of the previous weight update value in the weight is to bring in the fraction of the previous weight update value into the current weight to smooth out the new weight value.

To have a significant impact on the weight update value, the magnitude of gradient and the previous weight update value are controlled by learning parameters such as learning rate and momentum. As in two-termBP, the values of learning rate and parameters are acquired through experiments.The correct tuning of learning a parameter's value can assist in obtaining the correct value and sign of weight update. Afterwards, it effects the calculation of the new value of weight by using the weight adjustment method in Equation (30).

In experiments,a few authors have observed that the temporal behaviour of the gradient does make a contribution to the learning. This temporal behaviour can be seen as the changing of the gradient's sign during its movement on the error surface. Owing to the curvature on the error surface, the gradient behaves differently under certain conditions. In [6] stated that when the gradient possesses the same sign in several consecutive iterations, it indicates that the minimum lies ahead but when the gradient changes its sign in several consecutive iterations, it indicates that the minimum has been passed. By using this information, we can improve our wayto update the weight.

http://dx.doi.org/10.5772/51776

To sum up, based on the heuristic that has been discussed, the value of weight should be increased when the several consecutive signs of gradient remain the same and decreased if the opposite condition occurs. However, that is not the sole determination of the sign. Other factors such as LR,gradient, momentum, previous weight value and current weight value also play a greater role in affecting the weight changing sign and magnitude. The heuristic given previously is merelyan indicator and information about gradient movement on the surface and the surface itself by which we get a better understanding about the gradient and the error surface so that we can arrive at the enhancement on gradient movement in the weight space.

Through a thorough study on the weight change sign processon both algorithms, it can be concluded that the information about the sign of past and current gradient is very helpful in guiding us to improve the training performance which in this case refers to the movement of gradienton the error surface. However, factors like gradient, learning rate, momentum, previous weight update value and current weight value do have greater influence on the sign and magnitude of the new value of weight. Besides that, the assignment of initial weight value also needs to be addressed further.The negative as sign one at fifth iteration. value of weight may lead the weight to remain negative on consecutive occasions when the gradient possesses a negatives ign. As a result the node will not contribute much to the learning and output. This can be observed from the weight update equation as follows.

$$\Delta W(t) = \eta(t) \nabla E(W(t)) + \beta(t) \Delta W(t - 1) \tag{31}$$

At the initial training iteration,the errort end stobe large and so does the gradient. This large value of gradient together with its sign will dominate the weight update calculation and thus will bring a large change to the sign and magnitude of the weight. The influence of the gradient through the weight update value can be seen from Equation (31) and the weight adjustment calculation below.

$$W(t + 1) = \Delta W(t) + W(t) \tag{32}$$

If the value of initial weight is smallert han the gradient, the new weight update value will be more likely affected by the gradient. As a result, the magnitude and sign of the weight will be changed according to the fraction of the gradient since at this initial iteration,the previous weight update value is set to 0 and leaves the gradient to dominate the computation of weight up date value as shown by Equation (31). The case discussed before can be viewed in this way. Assume that the random function assigns a negative value for one of the weight connections at the hidden layer. Afterperforming the feed-forward process, the difference between the output at output layer with the desired output is large. Thus, theminimization method is performed by calculating the gradient with respect to the weight at hidden layer. Since the error is large, the value of the computed gradient becomes large also. This can be seen in the equation below.

$$\delta_k = -(t_k - O_k) \bullet O_k(O_k - 1) \tag{33}$$

$$\nabla E(W(t)) = \left(\sum_k \delta_k W_{kj}\right) O_j(1 - O_j) \tag{34}$$

Assume that from Equation (34), the gradient at hidden layer has a large positive value and since it is at the initial state, the previous weight update value is set to 0 and according to Equation (31), it does not contribute anything to the weight update computation. As the result, the weight update value is largely influenced by the magnitude and sign of the gradient (including the contribution of the learning rate in adjusting the step size of the gradient). By performing the weight adjustment method described by Equation (32), the value of weight update which is mostly affected by gradient will dominate the change of weight magnitude and sign. However, this can be applied also to weight adjustment in the middle of training, where the gradient and error are still large and the previous weight update value is set to a certain amount.

We can see that the large value of error will affect the value of gradient where it will be assigned with a relatively large value. This value will be used to fix the weight in the network by propagating back the error signal to the network. As a result, the value and the sign of weight will be adjusted to compensate the error. Another notice able conclusion attained from the experiment is that when for consecutive iterations the gradient retains the same signs,the weight value over the iterations is in creased while when the gradient changes its signs for several consecutive iterations, the weight value is decreased. However,the change of weight sign and magnitude is still affected by the parameters and factors included in Equation (33) and (34) as explained before.

The following examples show the change of weight affected by the sign of gradient at consecutive iterations and its value.The change of the weight is represented by a Hinton diagram. The following Hinton diagram is the representation of the weights in standard BP with adaptive learning network on a balloon dataset.

The Hinton diagram in Figure 3 illustrates the sign and magnitude of all weights connection between hidden and input layer as well as hidden and output layer at first iteration where the light colour indicates the positive sign of weight and the dark colour indicates the negative sign of weight.The size of the rectangle indicates the magnitude of the weight. The fifth rectangles in Figure 3 are the bias connection between the input layer to the hidden layer; however, the bias connection to the first hidden layer carries a small value so that its representation is not clearly shown in the diagram and its sign is negative. The biases have the value of 1. The resulting error at the first iteration is still large, which is 0.1243.

The error decreases gradually from the first iteration until the fifth iteration. The changes on the gradient are shown in the table below.

http://dx.doi.org/10.5772/51776

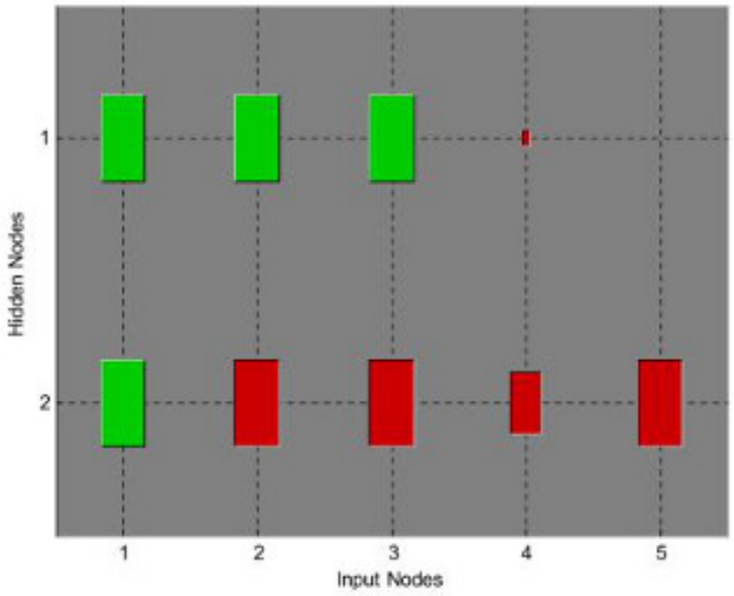

Figure 3. Hinton diagram of all weights connection between input layer and hidden layer at first iteration on Balloon dataset

		Input node 1	Input node 2	Input node 3	Input node 4	Bias
Iteration 1	Hidden node1	0.004	0.0029	0.0019	0.004	0.0029
	Hidden node2	-0.007	-0.005	-0.0033	-0.0068	-0.005
Iteration 2	Hidden node1	0.0038	0.0027	0.0017	0.0039	0.0025
	Hidden node2	-0.0063	-0.0044	-0.0028	-0.0062	-0.0042
Iteration 3	Hidden node1	0.0037	0.0025	0.0015	0.0039	0.0022
	Hidden node2	-0.0053	-0.0037	-0.0022	-0.0055	-0.0032
Iteration 4	Hidden node1	0.0035	0.0023	0.0012	0.0038	0.0016
	Hidden node2	-0.0043	-0.0029	-0.0016	-0.0046	-0.0021
Iteration 5	Hidden node1	0.0031	0.0019	0.0009	0.0036	0.0009
	Hidden node2	-0.0032	-0.0021	-0.0009	-0.0037	-0.001

Table 1. The gradient value between input layer and hidden layer at iterations 1 to 5.

		Hidden node 1	Hidden node 2	Bias
Iteration 1	Output node	0.0185	0.0171	0.0321
Iteration 2	Output node	0.0165	0.0147	0.0279
Iteration 3	Output node	0.0135	0.0117	0.0222
Iteration 4	Output node	0.0097	0.008	0.0151
Iteration 5	Output node	0.0058	0.0042	0.0078

Table 2. The gradient value between hidden layer and output layer at iterations 1 to 5.

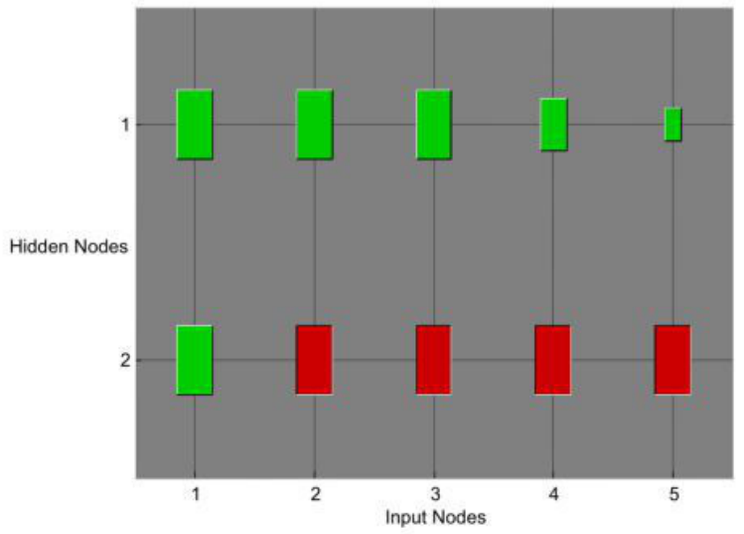

Figure 4. Hinton diagram of weightconnections between inputlayer and hidden layer at fifth iteration.

From the table above we can infer that the gradient in the hidden layer moves in a different direction while the one in the output layer moves in the same direction. Based on the heuristic,when each gradient moves in the same iteration for these consecutive iterations, the value of weight needs to be increased. However, it still depends on the factors that have been mentioned before. The impact on weight is given in the diagram below.

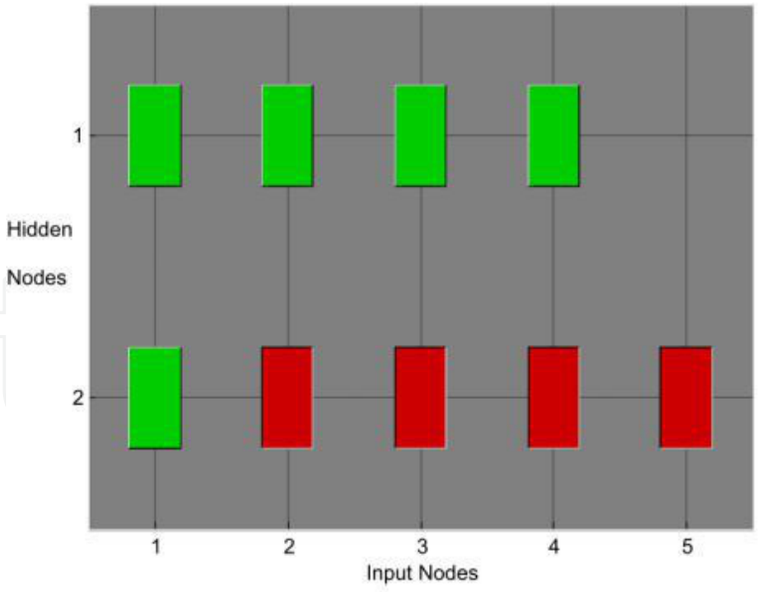

Figure 5. Hinton diagramof weight connections between inputlayer and hidden layer at 12th iteration.

Comparing the value of the weight in the fifth iteration with the first iteration, we can infer that most of the magnitude of weight on the connection between input and hidden layers in the fifth iteration becomes greater compared with that inthe first iteration. This

http://dx.doi.org/10.5772/51776

shows that the value of weight is increased over iterations according to the sign of gradient that is similar over several iterations. However, it is noticeable that the sign of the weight between input node 4 and the first hidden node as well as bias at input layer to first hidden node changes. This is due to the influence of the result of the multiplication of large positive gradient and LR which dominates the weight update calculation, and hence increases its magnitude and switches the weight direction. As a result, the error decreases to 0.1158 from 0.1243.

Atiterations 12 and 16 the error gradient moves slowly along the shallow slope in the same direction and brings smaller changes in gradient, weight and of course error itself. The change in the gradient is shown in the table below.

It can be seen from the sign of the gradient that it differs from the one at iterations 1-5, which means that the gradient at bias connection moves in a different direction. The same thing happens with the gradient at the third node of input layer to all hidden nodes where the sign changes from the one at fifth iteration. Another change occurs at the gradient in second input node to the first hidden node at iteration 16.

		Input node 1	**Input node 2**	**Input node 3**	**Input node 4**	**Bias**
Iteration 12	Hidden node1	0.0017	0.0007	-0.0003	0.0026	-0.0013
	Hidden node2	-0.0017	-0.001	0.0001	-0.0025	0.0006
Iteration 13	Hidden node1	0.0014	0.0005	-0.0006	0.0024	-0.0017
	Hidden node2	-0.0016	-0.001	0.0003	-0.0025	0.0009
Iteration 14	Hidden node1	0.0011	0.0003	-0.0009	0.0022	-0.0021
	Hidden node2	-0.0015	-0.0009	0.0005	-0.0025	0.0013
Iteration 15	Hidden node1	0.0008	0.0001	-0.0012	0.0021	-0.0028
	Hidden node2	-0.0012	-0.0008	0.0009	-0.0025	0.0018
Iteration 16	Hidden node1	0.0005	-0.0001	-0.0016	0.0019	-0.0035
	Hidden node2	-0.001	-0.0007	0.0013	-0.0025	0.0025

Table 3. The gradient value between input and hidden layer at iterations 12 to 16.

Owing to the change of gradient sign that has been discussed before, the change on weight at this iteration is clearly seen in its magnitude. The weights are large in magnitude compared with the one at fifth iteration since some of the gradients have moved in the same direction. The bias connection to the first hidden node is not clearly seen since its value is very small. However,its value is negative. For some of the weights, the changes in sign of the gradient have less effect on the new weight value since its value is very small. Thus, the sign of the weights remains the same. Moreover, although the gradient sign of the third input node and the first hidden node changes, the weight sign remains the same since the positive weight updateat the previous iteration and the changes of gradient are very small. Thus, it has a smaller impact on the change of weight although the weight update value is negative or decreasing. At this iteration, the error decreases to 0.1116.

Besides the magnitude change,the obvious change is seen in the sign of weight of the first input node to the second hidden node. It is positive at the previous iteration, but now it turns to negative.

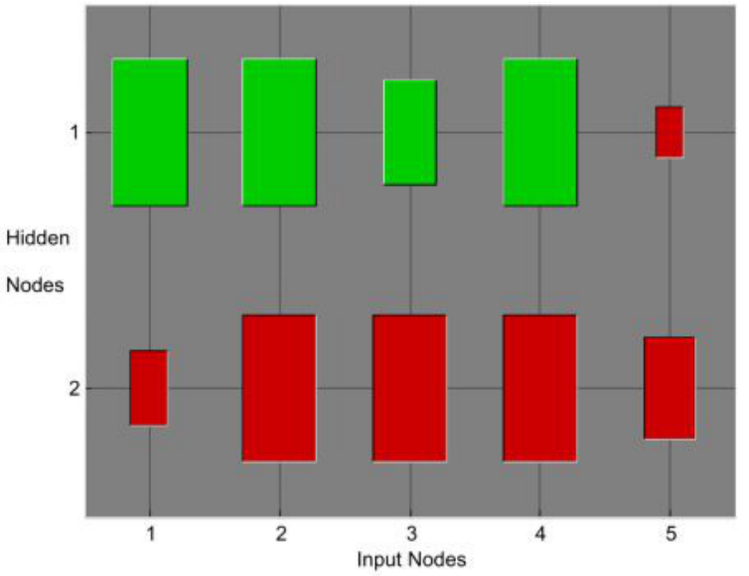

Figure 6. Hinton diagram of weight connections between input layer and hidden layer at 14th iteration.

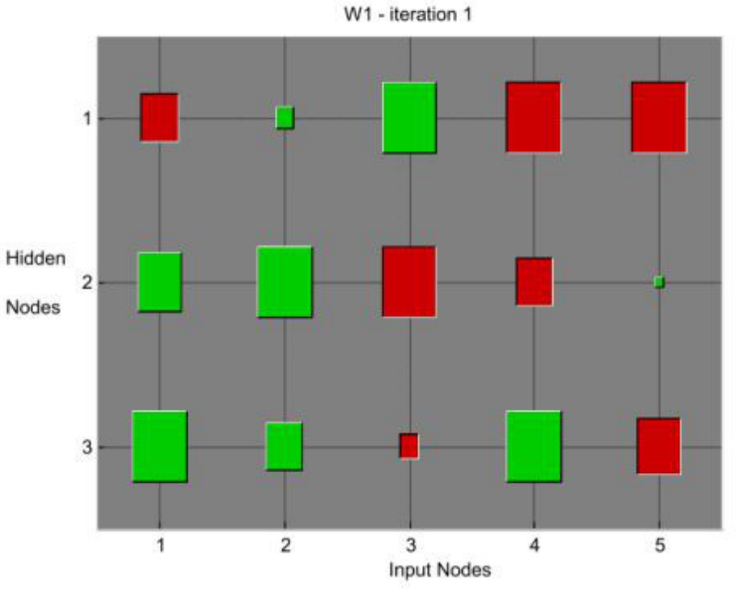

Figure 7. Hinton diagram of weight connections between input layer and hidden layer at 1st iteration.

From the experiment, the magnitude of this weight decreases gradually in small numbers over several iterations. This is due to the negative value of gradient at the first iteration.

http://dx.doi.org/10.5772/51776

Since its initial value is quite big, thus, the decrement does not have a significant effect on the sign. Moreover, since its value is getting smaller after several iterations, the impact of the negative gradient can be seen in the change of the weight sign. The error at this iteration decreases to 0.1089.

The next example is the Hinton diagram representing weights in standard BP with fixed parameters network on Iris dataset.

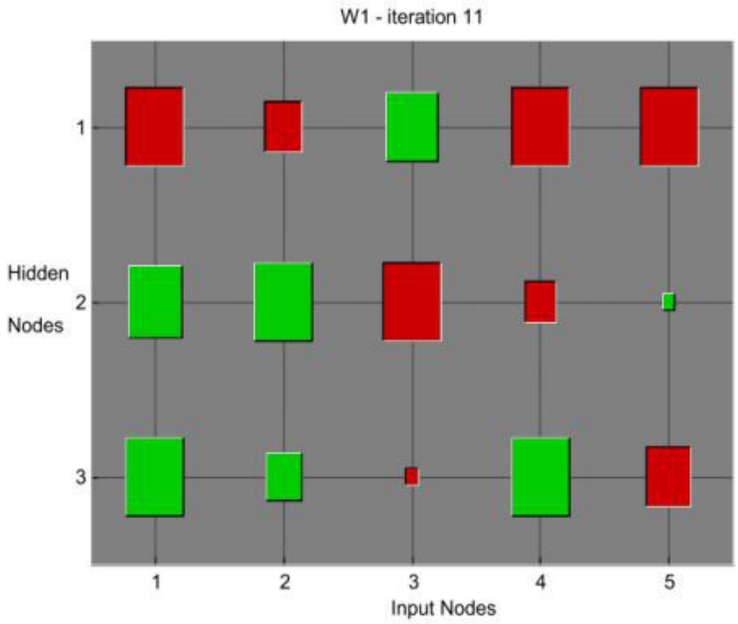

Figure 8. Hinton diagram of weight connections between input layer and hidden layer at 11th iteration.

		Input Node 1	Input Node 2	Input Node 3	Input Node 4	Bias
Iteration 1	Hidden Node 1	-0.018303748	-0.011141395	-0.008590033	-0.002383615	-0.003356197
	Hidden Node 2	0.001416033	0.000224827	0.002969737	0.001394722	2.13E-05
	Hidden Node 3	-0.001842087	-0.001514775	0.000163215	0.000251155	-0.000430432
Iteration 2	Hidden Node 1	-0.01660207	-0.010163722	-0.007711067	-0.002135194	-0.00304905
	Hidden Node 2	0.001824512	0.000525327	0.003040339	0.001392819	0.000106987
	Hidden Node 3	-0.001355907	-0.001195881	0.000330876	0.000285222	-0.000335039
Iteration 3	Hidden Node 1	-0.014965571	-0.009227058	-0.006852751	-0.001889934	-0.002755439
	Hidden Node 2	0.002207694	0.000812789	0.003095022	0.001386109	0.000188403
	Hidden Node 3	-0.000932684	-0.000914026	0.000466302	0.000309974	-0.000251081
Iteration 4	Hidden Node 1	-0.013416363	-0.008340029	-0.006036264	-0.00165541	-0.002478258
	Hidden Node 2	0.002525449	0.001063613	0.003111312	0.001367368	0.000258514
	Hidden Node 3	-0.000557761	-0.000660081	0.000576354	0.000327393	-0.000175859

Table 4. The gradient value between input and hidden layer at iterations 1-4

		Input Node 1	Input Node 2	Input Node 3	Input Node 4	Bias
Iteration 11	Hidden Node 1	-0.006205968	-0.004164878	-0.002294527	0.000583307	-0.001184916
	Hidden Node 2	0.003261605	0.001956214	0.00242412	0.000995646	0.000485827
	Hidden Node 3	0.000946822	0.00043872	0.00083246	0.000312944	0.000141871
Iteration 12	Hidden Node 1	-0.00559575	-0.003804835	-0.001989824	-0.000497406	-0.001074373
	Hidden Node 2	0.003205306	0.001986585	0.002247662	0.000920457	0.000488935
	Hidden Node 3	0.001041566	0.000519475	0.000821693	0.000299802	0.000164175
Iteration 13	Hidden Node 1	-0.005055156	-0.003484476	-0.001722643	-0.000422466	-0.000976178
	Hidden Node 2	0.003122239	0.001999895	0.002060602	0.00084271	0.000486953
	Hidden Node 3	0.001115805	0.000586532	0.000804462	0.000285504	0.000182402
Iteration 14	Hidden Node 1	-0.004575005	-0.003198699	-0.001487817	-0.000356958	-0.000888723
	Hidden Node 2	0.003016769	0.001998522	0.001865671	0.000763285	0.000480617
	Hidden Node 3	0.001172258	0.000641498	0.000782111	0.000270422	0.000197051

Table 5. The gradient value between input and hidden layer at iterations 11-14

Atiteration 11 ,the most obvious change in sign is on the weight between the second input node and the first hidden node. Based on the table of gradient, we might know that the gradient at this connection moves in the same direction through out iterations 1-4 and 11-14. However, due to the negative value of the gradient, the weight update value carries a negative sign that causes the value to decrease until its sign is negative. At this iteration, the error value decreases.

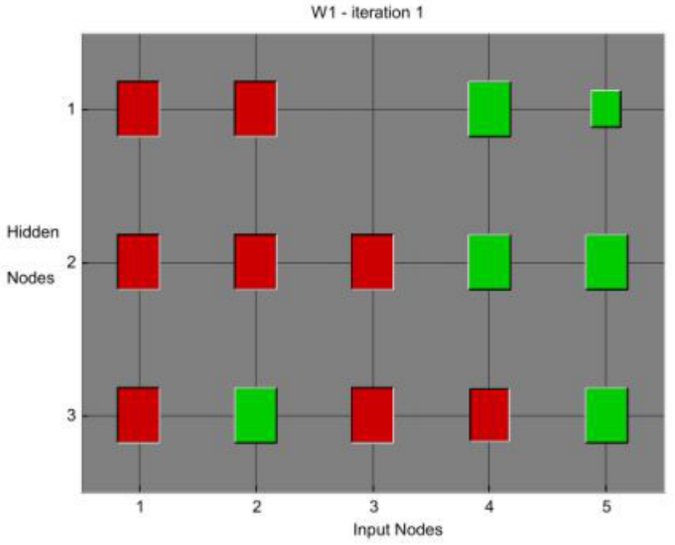

Figure 9. Hinton diagram of weight connections between input layer and hidden layer at 1st iteration.

http://dx.doi.org/10.5772/51776

The next example is the Hinton diagram representing weights in standard BPwith an adaptive learning parameter network on Iris dataset.

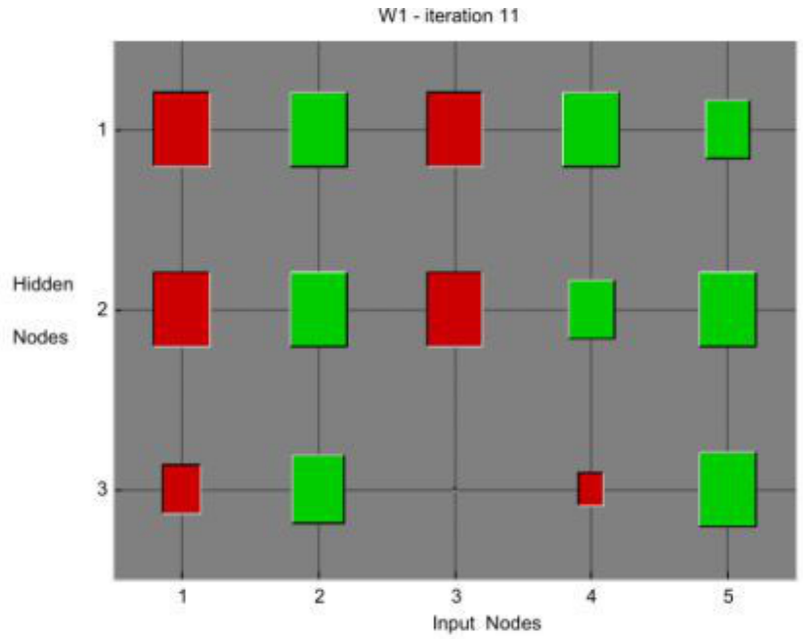

Figure 10. Hinton diagram of weight connections between input layer and hidden layer at 11th iteration.

		Input Node 1	Input Node 2	Input Node 3	Input Node 4	Bias
Iteration 1	Hidden Node 1	-0.002028133	0.001177827	-0.006110183	-0.002515178	6.16E-05
	Hidden Node 2	0.00327263	0.002733914	-0.000166769	-0.000344516	0.000749096
	Hidden Node 3	-0.004784699	-0.003919529	-0.000360459	0.000140134	-0.001022238
Iteration 2	Hidden Node 1	-0.002754283	0.000759663	-0.006493288	-0.00262835	-7.00E-05
	Hidden Node 2	0.003322116	0.002868889	-0.00034058	-0.000417257	0.000772938
	Hidden Node 3	-0.003433234	-0.003331085	0.000673172	0.000468549	-0.000802899
Iteration 3	Hidden Node 1	-0.002109624	0.001068822	-0.006027406	-0.002476693	3.47E-05
	Hidden Node 2	0.003299403	0.002898428	-0.00043867	-0.000457266	0.000775424
	Hidden Node 3	-0.002239138	-0.002627843	0.001246792	0.000621859	-0.000583171
Iteration 4	Hidden Node 1	-0.001150779	0.001525718	-0.005325608	-0.002246529	0.000189476
	Hidden Node 2	0.003268854	0.002945653	-0.000588941	-0.000519753	0.000780289
	Hidden Node 3	-0.000746758	-0.001700258	0.001868946	0.000774123	-0.000301095

Table 6. The gradient value between input and hidden layer at iterations 1-4 .

		Input Node 1	Input Node 2	Input Node 3	Input Node 4	Bias
Iteration 11	Hidden Node 1	0.001907574	0.002986261	-0.002973391	-0.001440144	0.00065675
	Hidden Node 2	0.000812331	0.002899602	-0.004647719	-0.002073105	0.000546702
	Hidden Node 3	-0.002190841	-0.000838811	-0.002332771	-0.000908119	-0.000280471
Iteration 12	Hidden Node 1	0.003681706	0.004088069	-0.002210031	-0.001245223	0.000986844
	Hidden Node 2	0.002483971	0.003929436	-0.003913207	-0.001884712	0.000855605
	Hidden Node 3	-0.002708979	-0.001126258	-0.002636196	-0.001002275	-0.000370089
Iteration 13	Hidden Node 1	0.00439905	0.004702491	-0.002255431	-0.001321819	0.001149609
	Hidden Node 2	0.002580744	0.004096122	-0.004087027	-0.00196994	0.000891341
	Hidden Node 3	-0.003069687	-0.001361618	-0.002782251	-0.001041828	-0.00043751
Iteration 14	Hidden Node 1	0.001502568	0.003228589	-0.004165255	-0.00193001	0.000663654
	Hidden Node 2	-0.000869554	0.002112189	-0.005859167	-0.002472493	0.00027103
	Hidden Node 3	-0.003177038	-0.001479172	-0.002727358	-0.001012298	-0.000466909

Table 7. The gradient value between input and hidden layer at iterations 11-12.

At the 11 th iteration, all weights change their magnitude and some have different signs from before. The weight between the second input node to the first and second hidden layers changes its sign to positive because of the positive incremental value since the gradient moves along the same direction over time. The positive incremental value gradually change the magnitude and sign of the weight from negative to positive.

7. Conclusions

This study is performed through experimental results achieved by constructing programs for both algorithms which are implemented on various datasets. The dataset comprises a small and medium dataset which will be broken down into two datasets, training and testing, with ratio percentages of 70% and 30% respectively. The result from both algorithms will be examined and studied based on its accuracy, convergence time and error. In addition, this study also studies the weight change sign with respect to the temporal behaviour of gradient to study the learning behaviour of the network and also to measure the performance of the algorithm. However, two-term BP with an adaptive algorithm works better in producing proper change of weight so that the time needed to converge is shorter compared with two-term BP without an adaptive learning method. This can be seen from the result of the convergence rate of the network. Moreover, the study on weight sign change of both algorithms shows that the gradient sign and magnitude and error have greater influence on the weight adjustment process.

http://dx.doi.org/10.5772/51776

Acknowledgements

Authors would like to thanks Universiti Teknologi Malaysia (UTM) for the support in Research and Development, and Soft Computing Research Group (SCRG) for the inspiration in making this study a success. This work is supported by The Ministry of Higher Education (MOHE) under Long Term Research Grant Scheme (LRGS/TD/2011/UTM/ICT/03 - VOT 4L805).

Author details

Siti Mariyam Shamsuddin*, Ashraf Osman Ibrahim and Citra Ramadhena

*Address all correspondence to: mariyam@utm.my

Soft Computing Research Group, Faculty of Computer Science and Information Systems, UniversitiTeknologi Malaysia, Malaysia

References

[1] Ng, S., Leung, S., & Luk, A. (1999). Fast convergent generalized back-propagation algorithm with constant learning rate. *Neural processing letters*, 13-23.

[2] Zweiri, Y. H., Whidborne, J. F., & Seneviratne, L. D. (2003). A three-term backpropagation algorithm. *Neurocomputing*, 305-318.

[3] Yu, C. C. B., & Liu, D. (2002). A backpropagation algorithm with adaptive learning rate and momentum coefficient. in 2002 International Joint Conference on Neural Networks (IJCNN 2002). May 12-May 17, 2002, *Honolulu, HI, United states: Institute of Electrical and Electronics Engineers Inc.*

[4] Dhar, V. K., et al. (2010). Comparative performance of some popular artificial neural network algorithms on benchmark and function approximation problems. *Pramana-Journal of Physics*, 74(2), 307-324.

[5] Hongmei, S., & Gaofeng, Z. (2009). A new BP algorithm with adaptive momentum for FNNs training. in 2009 WRI Global Congress on Intelligent Systems GCIS 2009. May 19, 2009-May 21, 2009. Xiamen, China: , *IEEE Computer Society.*

[6] Jacobs, R. A. (1988). Increased Rates of Convergence Through Learning Rate Adaptation. *Neural Networks*, 1(4), 295-307.

[7] Minai, A. A., & Williams, R. D. (1990). Back-propagation heuristics: A study of the extended Delta-Bar-Delta algorithm. in 1990 International Joint Conference on Neu-

ral Networks- IJCNN 90. June 17, 1990-June 21, 1990, *San Diego, CA, USA: Publ by IEEE.*

[8] Jin, B., et al. (2012). The application on the forecast of plant disease based on an improved BP neural network. in 2011 International Conference on Material Science and Information Technology, MSIT2011 September 16-September 18, 2011. Singapore, Singapore: Trans Tech Publications.

[9] Duffner, S., & Garcia, C. (2007). An online backpropagation algorithm with validation error-based adaptive learning rate. Artificial Neural Networks-ICANN 2007 , 249-258.

[10] Shamsuddin, S. M., Sulaiman, M. N., & Darus, M. (2001). An improved error signal for the backpropagation model for classification problems. *International Journal of Computer Mathematics*, 297-305.

[11] Iranmanesh, S., & Mahdavi, M. A. (2009). A differential adaptive learning rate method for back-propagation neural networks. World Academy of Science, Engineering and Technology , 38, 289-292.

[12] Li, Y., et al. (2009). The improved training algorithm of back propagation neural network with selfadaptive learning rate. in 2009 International Conference on Computational Intelligence and Natural Computing, CINC 2009. June 6-June 7, 2009Wuhan, China: IEEE Computer Society.

[13] Yamamoto, K., et al. (2011). Fast backpropagation learning using optimization of learning rate for pulsed neural networks. *Electronics and Communications in Japan*, 27-34.

[14] Hua, Li. C., Xiangji, J., & Huang, . (2012). Spam filtering using semantic similarity approach and adaptive BPNN. *Neurocomputing*.

[15] Xiaoyuan, L., Bin, Q., & Lu, W. (2009). A new improved BP neural network algorithm. in 2009 2nd International Conference on Intelligent Computing Technology and Automation, ICICTA 2009. October 10-October 11, 2009., *Changsha, Hunan, China: IEEE Computer Society.*

[16] Yang, H., Mathew, J., & Ma, L. (2007). Basis pursuit-based intelligent diagnosis of bearing faults. *Journal of Quality in Maintenance Engineering*, 152-162.

[17] Fukuoka, Y., et al. (1998). A modified back-propagation method to avoid false local minima. *Neural Networks*, 1059-1072.

[18] Riedmiller, M. (1994). Advanced supervised learning in multi-layer perceptrons-from backpropagation to adaptive learning algorithms. *Computer Standards and Interfaces*, 16(3), 265-278.

http://dx.doi.org/10.5772/51776

[19] Sidani, A., & Sidani, T. (1994). Comprehensive study of the back propagation algorithm and modifications. in Proceedings of the 1994 Southcon Conference,. March 29-March 31, 1994Orlando, FL, USA: IEEE.

[20] Samarasinghe, S. (2007). Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition. *Auerbach Publications.*

IntechOpen

IntechOpen