A new learning rate based on Andrei method for training feed-forward artificial neural networks

Khalil K. Abbo¹, Hassan H. Abrahim², Firdos A. Abrahim³

¹ Department of Mathematics, College of Computer Sciences & Mathematics, University of Mosul, Mosul, Iraq
² Department of Mathematics, College of Computer Sciences & Mathematics, University of Tikrit, Tikrit, Iraq
³ Department of Mathematics, College of Education, University of Tikrit, Tikrit, Iraq

Abstract
In this paper we developed a new method for computing learning rate for Back-propagation algorithm to train a feed-forward neural networks. Our idea is based on the approximating the inverse Hessian matrix for the error function originally suggested by Andrie. Experimental results show that the proposed method considerably improve the convergence rate of the Back-propagation algorithm for the chosen test problem.

1. Introduction
Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological neurons systems. As in nature, the network function is determined largely by the connections between elements. We can train a neural network to perform a particular function by adjusting the values of the connections(weights), between elements, commonly neural networks are adjusted, or trained so that a particular input leads to as specific target output. The network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically many such input/target pairs are used in this supervised learning to train a network. Batch training of the network proceeds by making weight and bias changes based on an entire set (batch) of input vectors[6]. The batch training of the Multi-layer Feed-forward Neural network (MFF) can be formulated as a non-linear unconstrained minimization problem [8, 9] Namely.

\[
\min E(w_j), w_j \in R^* \quad (1)
\]

where E is the batch error measure defined as the sum of squared differences Error functions over the entire training set, defined by

\[
E(w) = \frac{1}{2} \sum_{j=1}^{n} \sum_{p=1}^{M} (o_{j,p} - t_{j,p})^2 \quad (2)
\]

Where \((o_{j,p} - t_{j,p})^2\) is the squared differences between the actual j-th output layer neuron for pattern P and the target output value. The scalar P is an index over input-output pairs, the general purpose of the training is to search an optimal set of connection weights in the manner that the error of the network output can be minimized.

The most popular training algorithm is the Classical Batch Back Propagation (CBP) introduced by Rumelhart, Hinton and Williams[12]. Although the CBP algorithm is a simple learning algorithm for training Multi-layer Feed-Forward MFF networks, unfortunately it is not based on a sound theoretical basis and is very inefficient and unreliable. One iteration of the CBP algorithm can be written

\[
w_{j+1} = w_j - \alpha_j g_j \quad (3)
\]

Where \(w_j\) is the vector of current weights and biases, \(g_j = V E(w_j)\) and \(\alpha_j\) is the learning rate, with CBP the learning rate is held constant throughout training. The performance of the algorithm is very sensitive to the proper setting of the learning rate [5]. In order to overcome to the drawbacks of the CBP algorithm many gradient based training algorithms have been proposed in the literature [1, 2, 5, 7, 13].

2. Some Modifications on CBP.
A surprising result was given by Brazilian and Brownie [3], which gives formula for the learning rate \(\alpha_j\) and leads to super linear convergence. The main idea of Brazilia and Brownie (BB) method is to use the information in the previous iteration to decide the step size (learning rate) in the current iteration. The iteration in equation (3) is viewed as

\[
w_{j+1} = w_j - D_j s_j \quad (4)
\]

Where \(D_j = \alpha_j I\). In order to force matrix \(D_j\), having certain quasi-Newton(QN) property, is reasonable to require either

\[
\min \|D_j s_j - y_j\| \quad (5)
\]

Or

\[
\min \|D_{j-1} s_{j-1} - y_{j-1}\| \quad (6)
\]

Where \(s_{j-1} = w_{j-1} - w_j \) and \(y_j = g_j - g_{j-1}\). Solving equation (5) or (6) for \(\alpha_j\) we get the following formulas

\[
\alpha_{j+1}^{BB1} = \frac{s^T_j y_j}{y_j y_j} \quad (7)
\]

\[
\alpha_{j+1}^{BB2} = \frac{s^T_j s_j}{s_j y_j} \quad (8)
\]

respectively. Note that we abbreviate the method defined in equation(3) with learning rate defined in equations (7) and (8) as BB1 and BB2 methods, respectively.

An alternative approach is based on the work of Plagianakos et al [11].

Following this approach, equation (3) is reformulated to the following Scheme:

109
$$w_{k+1} = w_k - B g_k$$  \hspace{1cm} (9)

Where $B = \text{diag} \{ \lambda_1, \lambda_2, \ldots, \lambda_n \}$ and $\lambda_i, i = 1, \ldots, n$ are eigen values for the $\nabla^2 E(w_k)$, or approximations to the Eigen-values of $\nabla^2 E(w_k)$. A well known difficulty to this approach is that the computation of the Eigen values or estimating them is not a simple task, hence the schema defined in equation (9) is not practical.

3- Development Method

In the following we suggest another procedure for computing a scalar approximation of the Hessian of the function $E$ at $w_k \in \mathbb{R}^n$ which can be used to get the step-size along the negative gradient. Let us consider the initial point $w_0$ where $E(w_0)$ and $g = \nabla E(w_0)$ can immediately be computed. Using the backtracking procedure (initialized with $\alpha_0 = 1$) we can compute the step-length

$$\alpha_1 = \arg \min_{\alpha \in (0, 1)} E(w_k - \alpha g_k)$$

So, the first step is computed using the backtracking along the negative gradient. Now, at point $w_{k+1} = w_k - \alpha_k g_k$, $k = 0, 1, \ldots$, from Taylor series we have

$$E_{k+1} = E_k - \alpha_k g_k^T g_k + \frac{1}{2} \alpha_k^2 g_k^T \nabla^2 f(z) g_k,$$  \hspace{1cm} (10)

Where $z$ is the line segment connecting $w_k$ and $w_{k+1}$. Having in view the local character of the searching procedure and that the distance between $w_k$ and $w_{k+1}$ is enough small we can choose $z = w_k$ and consider $\alpha_{k+1}$ as a scalar approximation of the $\nabla^2 E(w_{k+1})$, where $\alpha_{k+1} \in \mathbb{R}$. This is an anticipative view point, in which a scalar approximation to the Hessian at point $w_{k+1}$ is computed using only the local information from two successive points: $w_k$ and $w_{k+1}$. There for, we can write see [4]:

$$\alpha_{k+1} = \frac{2}{g_k^T g_k} [E_{k+1} - E_k + \alpha_k g_k^T g_k]$$  \hspace{1cm} (11)

Observe that at $w_{k+1}$, $k = 0, 1, \ldots$ we know $E(w_{k+1})$, $g_{k+1}$ and approximation of $\nabla^2 E(w_{k+1})$ as $\alpha_{k+1}$. Now, in order to compute the next estimation $w_{k+2} = w_{k+1} - \alpha_{k+1} g_{k+1}$, we must consider a procedure to step length $\alpha_{k+1}$. For this let us consider the function:

$$\phi_{k+1}(\alpha) = E(w_{k+1}) - \alpha g_{k+1}^T g_{k+1} + \frac{1}{2} \alpha^2 g_{k+1}^T g_{k+1}$$  \hspace{1cm} (12)

Observe that $\phi_{k+1}(\alpha) = E(w_{k+1})$, $\phi_{k+1}(\frac{2}{\alpha_{k+1}}) = E(w_{k+1})$, and $\phi'(\alpha) = -g_{k+1}^T g_{k+1} < 0$. Therefore, $\phi_{k+1}(\alpha)$ is a convex function for all $\alpha \geq 0$. To have a minimum for $\phi_{k+1}(\alpha)$ we must have $\alpha_{k+1} > 0$. Considering for moment $\alpha_{k+1} > 0$, then from $\phi_{k+1}''(\alpha)$ we get

$$\frac{1}{\alpha_{k+1}} = \frac{1}{E_{k+1}},$$  \hspace{1cm} (13)

as the minimum point of $\phi_{k+1}(\alpha)$, observe that

$$\phi_{k+1}'(\alpha) = E_{k+1} - \frac{1}{2 \alpha_{k+1}} \| g_k \|$$  \hspace{1cm} (14)

Showing that, if $\alpha_{k+1} > 0$, then at every iteration the value of function $E_{k+1}$ is reduced[4]. On the other hand, if happen that $\alpha_{k+1} < 0$, we may define the following correction on the value of $\alpha_{k+1}$

$$\alpha_{k+1} = \frac{2}{g_k^T g_k} [E_{k+1} - E_k + \alpha_k g_k^T g_k + \delta], \quad \delta > 0$$  \hspace{1cm} (15)

$$\alpha_{k+1} = \frac{2}{g_k^T g_k} [E_{k+1} - E_k + (\alpha_k + \eta_k) g_k^T g_k]$$  \hspace{1cm} (16)

To avoid the above problem (equations 15 and 16), we suggest the following formula to compute the learning rate at each epoch

$$\alpha_k = \frac{5 \alpha_1 \| g_k \|}{\| g_k \| + \alpha_1}, \quad k = 1$$  \hspace{1cm} (17)

$$\alpha_k = \frac{5 \alpha_1 \| g_k \|}{\| g_k \| + \alpha_1}, \quad k = 1$$

with the use of Backtracking strategy to achieve the Wolfe conditions.

Algorithm(PISBP).

Step1. Initialization: Select $w_0 \in \mathbb{R}^n$, $\epsilon > 0$ and $0 < \rho < \sigma < 1$. Compute $E(w_0)$ and $g_0 = \nabla E(w_0)$. Consider $d_0 = -g_0$, and set $k = 1$.

Step2. Test for continuation of iterations. IF $\| g_k \| < \epsilon$, then stop. Else go to Step 3.

Step3. Learning rate computation. Compute $\alpha_k$ from (17) and test for the Wolfe line Search conditions and update the variables

$$w_{k+1} = w_k - \alpha_k g_k$$

Compute $E_{k+1}$, $g_{k+1} = w_{k+1} - w_k$, and $\gamma_k = g_{k+1} - g_k$.

Step4. Set $k = k+1$ and go to Step 2.
4. Experiments and Results:
A computer simulation has been developed to study the performance of the following algorithms.
1- GD: classical back-propagation algorithm.
3- F1SBP: New suggested training algorithm.

The simulations have been carried out using MATLAB(7.6) the performance of the MSBP has been evaluated and compared with batch versions of the above algorithm. The algorithms were tested using the initial weights, initialized by the Nguyen – widrow method [10] and received the same sequence of input patterns . The weights of network are updated only after the entire set of patterns to be learned has been presented . For each of the test problems, a table summarizing the performance of the algorithms for simulations that reached solution is presented . The reported parameters are min the minimum number of epochs for 50 simulation , mean the mean value of epochs for 50 simulation, Max the maximum number of epochs for 50 simulation, Tav the average of total time for 50 simulation and Succ, the succeeded simulations out of (50) trails within error function evaluations limit. If an algorithm fails to converge within the above limit considered that it fails to train the FFNN, but its epochs are not included in the statical analysis of the algorithm, one gradient and one error function evaluations are necessary at each epoch.

1- Problem (XOR Problem)
The first problem we have been encountered with is the XOR Boolean function problem, which is considered as a classical problem for the FFNN training . The XOR function maps two binary inputs to a single binary output. As it is well known this function is not linearly separable. The network architectures for this binary classification problem consists of one hidden layer with 3 neurons and an output layer of one neuron. The termination criterion is set to $e_t \leq 0.002$ within the limit of 1000 epochs, and table(1) summarizes the result of all algorithms i.e for 50 simulations the minimum epochs for each algorithm are listed in the first column (Min), the maximum epochs for each algorithm are listed in the second column, third column contains (Mean) the mean value of epochs and (Tav) is the average of time for 50 simulations and last columns contain the percentage of succeeds of the algorithms in 50 simulations.

Table (1): Results of simulations for the XOR function
<table>
<thead>
<tr>
<th>algorithms</th>
<th>min</th>
<th>max</th>
<th>mean</th>
<th>Tav</th>
<th>succ</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>230</td>
<td>2000</td>
<td>652.78</td>
<td>8.83028</td>
<td>92%</td>
</tr>
<tr>
<td>GDA</td>
<td>50</td>
<td>84</td>
<td>66.5</td>
<td>1.13614</td>
<td>100%</td>
</tr>
<tr>
<td>F1SBP</td>
<td>3</td>
<td>26</td>
<td>8.9</td>
<td>0.5456</td>
<td>100%</td>
</tr>
</tbody>
</table>

2- Function Approximation Problem.
The second problem we have considered is the approximation of continuous function, $f(x) = \cos(\pi x) + 0.1 \cdot \text{rand} \cdot (\sin x)$
Where $x = -1:0.05:1$. This problem maps one real input to a single real output. The selected architecture of the FFNN is one neuron in input lager, ten neuron in hidden layer and one neuron in output neuron, with sigmoid function in hidden neuron's and a linear function in output neuron. The error goal has been let to 0.001 and the maximum epochs to 1000. The results of the simulations presented in table (2) and figure(2)

Table(2): Results of simulations for the Function Approximation Problem
<table>
<thead>
<tr>
<th>algorithms</th>
<th>min</th>
<th>max</th>
<th>mean</th>
<th>Tav</th>
<th>succ</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>GDA</td>
<td>191</td>
<td>795</td>
<td>391.94</td>
<td>5.60024</td>
<td>100%</td>
</tr>
<tr>
<td>F1SBP</td>
<td>107</td>
<td>662</td>
<td>285.54</td>
<td>8.29016</td>
<td>100%</td>
</tr>
</tbody>
</table>

3- SPECT Heart Problem.
This dataset contains data instances derived from cardiac single proton Emission Computed Tomography (SPECT) images from the University of Colorado. This is also a binary classification task, where patients heart images are classified as normal is abnormal. The class distribution has 55 instances of the abnormal class 20.6% and 212 instances of the normal class (79.4%)
From them there have been selected 80 instances for the training process and the remainder 187 for testing the neural networks generalization capability. The network architecture for this medical classification problem constitute of 1 hidden layer with 6 neurons and an output layer of 2 neurons. The termination criterion is set to $E_{\infty} \leq 0.1$ within the limit of 1000 epochs.

Table(3): Results of simulations for the SPECT Heart Problem
<table>
<thead>
<tr>
<th>algorithms</th>
<th>min</th>
<th>max</th>
<th>mean</th>
<th>Tav</th>
<th>succ</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>505</td>
<td>1827</td>
<td>989.9</td>
<td>13.14658</td>
<td>100%</td>
</tr>
<tr>
<td>GDA</td>
<td>70</td>
<td>525</td>
<td>190.82</td>
<td>2.71288</td>
<td>100%</td>
</tr>
<tr>
<td>F1SBP</td>
<td>25</td>
<td>87</td>
<td>55.52</td>
<td>2.1963</td>
<td>100%</td>
</tr>
</tbody>
</table>

5. Conclusions
In this paper we proposed a new formula for computing learning rate in the back-propagation algorithm for training feed-forward multi-layer neural networks. Based on our numerical experiments, we concluded that our proposed method outperforms classical Back-propagation and adaptive Back-propagation training algorithms and has a potential to significantly enhance the computational efficiency and robustness of training process.
References

Andrei معدل تعليم جديد باعتماد علي طريقة Andrei في تدريب الشبكات العصبية الاصطناعية ذات التغذية الأمامية
خليل خضر عبو1، حسن حسين إبراهيم2، فطون علي إبراهيم3
1قسم الرياضيات، كلية علوم الحاسب والرياضيات، جامعة الموصل، الموصل، العراق
2قسم الرياضيات، كلية علوم الحاسب والرياضيات، جامعة تكريت، تكريت، العراق
3قسم الرياضيات، كلية التربية، جامعة تكريت، تكريت، العراق

المختصر
تم في هذا البحث تطوير خوارزمية جديدة لحساب عامل التعلم للشبكات العصبية ذات التغذية الأمامية. تحتوي الفكرة على تعريف طريقة Andréi ونظام استخدام تقييم جديد لصنفية هرمية لدالة الخطأ. وقد بين النتائج العديدة أن الطريقة المفترضة تقارب إلى الحل بشكل أسرع من الطرق الأخرى ثم مقارنتها معها.