Iraqi Journal of Statistical Sciences (23) 2013
PP[1-10]

Improving the Learning Rate of the Back Propagation Algorithm by Aitkin Process

D. Khalil K. Abbo[*] Hind H. Mohammed[**]

تحسين عامل التعلم لـخوارزمية الانتشار العكسي باستخدام صيغة أيتكين المستخاص

تستخدم خوارزمية الانتشار العكسي Back propagation (BP) لتدريب الشبكات العصبية المتعددة الطبقات ذوات التغذية الأمامية (FFMNN) إلا أن هذه الخوارزمية غالباً ما تنحرف باتجاه قيمة صغرى محلية مما يؤدي إلى زيادة الوقت اللازم، ولكن الاختيار المناسب لعامل التعلم يقلص الوقت المطلوب للتدريب وبهذا نتغلب على التقارب البطـيء لخوارزميـة الانتشار العكسي (BP). تم في هذا البحث اشتقاق عامل تعلم جديـد لخوارزميـة الانتشـار العكسي (BP) باستخدام صيغة أيتكين حيث يتمتع عامل التعلم المقتـرح بميـزة اسـتخدامه للمشتقة الأولى فقط وبذلك يكون مناسباً جداً للشبكات الكبيرة ومجموعات التدريب ذوات البعد الكبير. تم اختبار كفاءة الصيغة المقترحة باستخدام مسائل الاختبار القياسـيةِ وهـي: دالـة الاختبار القياسية المتضمنة مسألة القلب (Heart Problem) ، ومسألة الـ(XOR) وأخيـرا مسألة تقريب الدوال (Function Approximation).

Abstract

The Back Propagation Algorithm is used for training feed Forward Multilayer Neural Networks (FFMNN).But often this algorithm takes long time to converge since it may fall into local minimu, for this reason we need a long time to train the network. The suitable choice of the learning rate helps us to escape from slow convergent for the BP and reduce the time of learning. In this paper, we derived a new adaptive learning rate for the BP algorithm, our derivation is based on the Aitkin's process. The most important distinct feature of our approach is the computing of the learning rate needs only first order derivatives and is suitable for large training sets and large networks. Its efficiency is proved on the standard test functions including heart , XOR and function approximation problems .

*Lecturer \ Department of Mathematics \ College of Computers Sciences and Mathematics, Mosul University.
**Assit. Lecturer \ Department of Mathematics \ College of Computers Sciences and Mathematics, Mosul University.

Received Date 13/6/2012 —— Accept Date 26/9/2012

1. Introduction

Methods to speed up the learning phase and to optimize the learning process in Feed Forward Neural Networks (FFNN) have been recently studied and several new adaptive learning algorithms have been discovered [Abbo&Zena(2011),Jarmo,et.al.(2003),Johansson,etal.(1990),Kostopoulos,et.at, Plagianakos,et.al.(1998), Sabeur & Farhat (2008), Zulhadi,etal.(2010)]. Some of them introduce the momentum term [Daniel,etal.(1997), Huajin,etal.(2011)], others use the alternative cost functions or dynamic adaptation of the learning parameters [Shahla,etal.(1997), Steven & Narciso(1999)]. Many apply special techniques of initialization of weights [Nguyen & Widrow(1990)].

Most of them apply the higher order gradient optimization routines to minimize the appropriately error function [Amir,et.al.(2005), Livieris&Pintelas(2008),Mollar(1993), Rumelhart,etal. (1986)], the multivariable function that depends on the weights of the network. However there is still the problem of accelerating the learning process, especially when large training sets and large network are used. The neural networks training can be formulated as minimization a non-linear unconstrained optimization problem [Livieris,et.at (2009)]. The energy or cost function to be minimized is defined in the usual way as the squared difference between the destination and the actual responses of the output neurons over all *P* training samples. Let us assume the multilayer FFNN of *N* input and *M* output neurons. The number of hidden layers may be arbitrary just as the number of neurons in the layers. The supervised learning of this net is equivalent to the minimization of the energy function (the error function), which can be written as follows:

$$E(w) = \frac{1}{2}\sum_{j=1}^{P}\sum_{i=1}^{M}(O_i^{(j)} - T_i^{(j)})^2 \qquad \text{...........} \quad (1)$$

The variables O_i and T_i stand for actual and desired response of i_th output neurons, respectively. The superscript denotes the particular learning pattern. The vector *w* is composed of all weights in the net.

Summation of the actual errors takes place over all *M* output neurons and all *P* learning data (*x*,*T*), where the *N*-dimensional vector *x* is the input vector and the *M*-dimensional vector *T* is the destination (Target) vector associated with *x*.

Back Propagation (BP) is a learning procedure that adjusts the weight vector w through a steepest descent with respect to *E* in weight space:

$$w_{k+1} = w_k - \gamma\, g_k \qquad \text{..........} \quad (2)$$

Where $g_k = \nabla E(w_k)$, γ is the learning rate which is constant $\gamma \in (0,1)$ and w_k is a vector representing the weights at iteration (epoch) step *k*. Though the procedure is widespread-dependent, disadvantages. First convergence is fast only if the parameter setting is nearly optimal. Furthermore, the convergence rate is slow (linear) and decreases rapidly as the problem size increases. Finally, convergence is guaranteed only if the learning rate γ is small enough [Kuan & Hornik (1991), Rumelhart, et.al.(1986)]. The main problem then is to determine a priori what small enough means. In other words, for shallow minimum the learning rate is often too small where as for narrow minimum it is often too large and the procedure never converges therefore the BP algorithm with constant learning rate (which is called classical BP i.e. CBP) tends to inefficient [Rumelhart, et.al. (1986.)].

The remainder of this paper is organized as follows. Section 2 presents a brief summary of Aitkin's process, section 3 presents the proposed BP algorithm (AIBP algorithm say). Section 4, reports our experimental results and in section 5 are presented our concluding remarks. We summarize the CBP algorithm as follows,

CBP Algorithm:

```
Step(1): initialization : Number of epochs, k=1, $\gamma \in (0,1)$ error goal=eg,
         stopping criteria $\varepsilon > 0$. Choose wk randomly and compute
         $g_k = \nabla E(w_k)$.
Step(2):check for convergence: If $\|g_k\| < \varepsilon$ or $E(w_k) < eg$
         stop $w_k$ is the optimal else go to step (3).
step(3): set $d_k = -g_k$ and update variable $w_k$ :
         $w_{k+1} = w_k + \gamma d_k$, set $\gamma=0.01$ and Compute $g_{k+1}$, $E_{k+1}$ .
Step(4): set $k = k+1$ and go to step (2)
```

2. Accelerated with Aitkin's Δ^2 process

When a sequence or an iterative process is slowly converging, a convergence acceleration process has to be used. It consists in transforming the slowly converging sequence into a new one which, under some assumptions converges faster to the same limit. Aitken's process is the most well known sequence transformation. It has been proved that able to accelerate the convergence linearly converging sequences [Clade & Michela (2007)].

Let $\{w_k\}^{\infty}$ be a linearly convergent sequence of values converging to be some paint w^* that is for $e_k = w_k - w^*$.

$$\lim \frac{|e_{k+1}|}{|e_k|} = \mu \le 1 \qquad \text{................} \quad (3)$$

To investigate the construction of a sequence $\{w_k\}^{\infty}$ which converges more rapidly to w^* [Clade & Michela (2007)]. Suppose that w_k generated by the equation (2), k=1,2,3 converges linearly so it satisfies :

$$w_{k+1} - w^* = \mu (w_k - w^*) \qquad \text{…………..} \quad (4)$$

and

$$w_{k+2} - w^* = \mu (w_{k+1} - w^*) \qquad \text{……………} \quad (5)$$

Solving equations (4) and (5) for w^* while eliminating μ leads to

$$w^* = w_k - \frac{(w_{k+1} - w_k)^2}{w_{k+2} - 2w_{k+1} + w_k} \qquad \text{……………} \quad (6)$$

In general, the original assumption (3) will not be true, nevertheless it is expected that the sequence $\{w_k\}^{\infty}$, defined by :

$$\overline{w_k} = w_k - \frac{(w_{k+1} - w_k)^2}{w_{k+2} - 2w_{k+1} + w_k} \qquad \text{…………….} \quad (7)$$

converges more rapidly to w^* than original sequence $\{w_k\}^{\infty}$. The point $\overline{w_k}$ is better approximation of w^* than w_k or w_{k+1} . The formula (7) can be written [3] in the equivalent form as :

$$w_i = w_k - \frac{(w_{k+1} - w_k) * (w_{k+1} - w_k)}{w_{k+2} - 2w_{k+1} + w_k} \qquad \text{…………….} \quad (8)$$

Where $i = k+3, k+4, \ldots$, we see that the formula (8) is suitable when the sequence w_k is real or complex numbers, for a vector sequence a scalar transformation could be used separately on each component or some modifications are made.

3. Proposed Learning Method

Derivation of the Method

In this section we present a modified Back Propagation (AIBP) algorithm by simple multiplicative modification of the learning rate. The idea is to modify the steepest descent method by introducing a relaxation of the following form :

$$w_{k+1} = w_k + \gamma \, \alpha_k \, d_k \qquad \text{……………} \quad (9)$$

where γ is the learning rate which is used in a classical BP and it has constant value, $\alpha k \in (0,1)$ is the relaxation parameter and $d_k = -g_k \, , \; \forall \, k$. To derive the value of αk , assume that $w_k, \; w_{k+1}$ are generated by the algorithm(CBP) and let $s_k = w_{k+1} - w_k$ and $y_k = g_{k+1} - g_k$. from equations (2) and (8) we have:

$$w_{k+1} - w_k = -\gamma \, g_k \qquad \text{…………..} \quad (10)$$

$$w_{k+2} - 2w_{k+1} + w_k = -\gamma \, y_k \qquad \text{……………} \quad (11)$$

use equations (10) and (11) in the equation (8) and multiply the numerator and denominator of the last term in (8) by y_k to get

$$: w_i = w_k + \frac{\gamma\ g_k^T g_k}{y_k^T y_k} y_k \qquad \ldots\ldots\ldots\ldots\ldots (12)$$

or $w_i = w_k + \gamma\,\alpha_k\, d_k$ .... (13)

Where

$$\alpha_k = \frac{g_k^T g_k}{y_k^T y_k} \qquad \ldots\ldots\ldots\ldots\ldots (14)$$

We see from equation (12), to compute w_i we need only two points namely w_k and w_{k+1}. Therefore our algorithm can be stated as:

Given w_k compute w_{k+1} using CBP then other points can be computed as: $w_{k+2} = w_{k+1} + \gamma\alpha_k d_{k+1}$. To insure the descent property for d we choose

$d_{k+1} = -g_{k+1}$ instead of $d_k = y_k$

We summarize our suggested algorithm (AIBP) as:

Step(1): Number of epochs *k=1* , $\gamma \in (0,1)$, error gol =eg, $\varepsilon > 0$,choose w_k and compute $g_k = \nabla E(w_k)$. Set $d_k = -g_k$, $\alpha_k = 1$.

Step(2): Test for convergence: If $E(w_k) < eg$ or $\|g_k\| < \varepsilon$ stop else go to step(3).

Step(3): Update the variables : If $k = 1$. $w_{k+1} = w_k + \gamma\, d_k$ and compute

$$g_{k+1}, E_{k+1}, y_k, \alpha_k = \frac{g_k^T\ g_k}{y_k^T\ y_k} \text{, and set } d_{k+1} = -g_{k+1}$$

else $w_{k+1} = w_k + \gamma\alpha_k\, d_k$ and compute

$$g_{k+1}, E_{k+1}, y_k, \alpha_k = \frac{g_k^T\ g_k}{y_k^T\ y_k}\ , \text{ and set } d_{k+1} = -g_{k+1}$$

Step(4); Set k=k+1 and go to step(2).

3.2 Convergence Analysis:

In general there is no an algorithm which is convergence in all cases , therefore in convergence analysis for algorithms often some mild assumptions are made. Under the following assumption we show that our algorithm (AIBP) is globally convergent:

1. Assume that the Error function is bounded below on the level set $S = \{w : E(w_k) \le E(w_1)\}$
2. $E(w)$ is convex function on the convex set S and $\alpha k \in (0,1)\ \forall$ k.

The following theorem (which is similar to one given in [Andrei (2005)] gives the convergence of AIBP algorithm.

Theorem(1):

Suppose that $E(w)$ is strongly convex on S, with $\nabla^2 E(W) \le M$ where M is a positive constant. If α_k given in (14) has an accumulation point $\alpha \in (0,1)$, then the sequence w_k generated by AIBP algorithm convergence linearly to w^*.

Proof:

Let us consider

$$\Phi_k(\alpha) = E(w_k - \gamma\alpha_k g_k)$$

Then by Taylor theorem we have:

$$E(w_k - \gamma\alpha_k g_k) = E_k - \gamma\alpha_k g_k^T g_k + \frac{1}{2}\ \gamma\,\alpha_k^2\ g_k^T \nabla^2 E_k g_k$$

Since E is assumed to be convex it follows that $\Phi(\alpha)$ is convex function and

$\Phi_k(0) = E_k$ and

$$E(w_k - \gamma\alpha_k\, g_k) \le E_k - (\alpha_k - \frac{M\gamma}{2}\alpha_k^2)\gamma\, g_k^T\, g_k$$

$(\alpha_k - \frac{M\gamma}{2}\alpha_k^2)$ is concave function on (0,2/Mγ) and has Max value at 1/ Mγ

therefore $E_{k+1} \le E_k \quad \forall\, k$

Since E is bounded below. It follows that

$\lim\limits_{k\to\infty} E_k - E_{k+1} = 0$

Since $\Phi(\alpha)$ is convex function with min value at point

$$\alpha_{\min} = \frac{g_k^T g_k}{\gamma\, g_k^T \nabla^2 E_k g_k} > 0$$

On the other hand $\Phi_k(0) = E_k$ and $\Phi(\alpha_r) = E_k$, where $\alpha_r = 2\alpha_{\min}$, But

$$\Phi(\alpha_{\min}) = E_k - \frac{(g_k^T g_k)^2}{g_k^T \nabla^2 E_k g_k} < E_k$$

Then for $\alpha \in [0, 2\,\alpha_{\min}], \Phi_k(\alpha) < \Phi_k(0)$

the reminder of the proof is similar to theorem 2 in [Andrei (2005)] hence is omitted . We conclude that

$$E(w_k) - E(w_{k+1}) \ge \frac{\gamma}{2M}\, g_k^T g_k$$

Since $E_k - E_{k+1} \to 0$ for large k therefore $g_k \to 0$

4. Experiments and Results:

A computer simulation has been developed to study the performance of the learning algorithms. The simulations have been carried out using MATIAB(7.6) the performance of the AIBP has been evaluated and compared with batch versions of the classical Back propagation (CBP) known as (traingd) see appendix, in the neural network toolbox, adaptive

BP (ABP) (traingda) . Toolbox default values for the heuristic parameters of the above algorithms are used unless stated otherwise. The algorithms were tested using the initial weights, initialized by the Nguyen –Widrow method [Nguyen & Widrow (1990)] 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, mean the mean value of epochs, Max the maximum number of epochs, Tav the average of total time 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 statistically analysis of the algorithm, one gradient and one error function evaluations are necessary at each epoch.

4.1 Problem (1): (Speect Heart Problem)

This data set contains data instances derived from Cardiac Single Proton Emission Computed Tomography (SPECT) images from the university of Colorado [Livieris, et.al (2009), Livieris, et.al (2011)]. The network architectures for this medical classification problem consists of one hidden layer with 3 neurons and an output layer of one neuron. The termination criterion is set to $E \leq 0.1$ within limit of 2000 epochs, table(1) summarizes the results of all algorithms i.e for 50 simulations the minimum epoch for each algorithm is listed in the first column (Min), the maximum epoch for each algorithm is 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 contains the percentage of succeeds of the algorithms in 50 simulations.

Table(1): Results of simulations for the Heart problem

Algorithms	Min	Max	Mean	Tav	Succ
CBP	1955	1955	1955	0.0901s	2 %
ABP	211	1596	680.47	0.032197s	98 %
AIBP	167	1207	500.88	0.024648s	68 %

Form table (1), we note that the algorithm ABP is the beast algorithm with respect to the succeeded simulations, while AIBP is the beast with respect to the epochs number and time.

4.2 Problem (2): Continuous Function Approximation:

The second test problem we consider is the approximation of the continuous trigonometric function: $f(x) = \sin(x) * \cos(3x)$.

The network architecture for this problem is 1-15-1 FNN (thirty weights, sixteen biases) is trained to approximate the function f(x), where $x \in [-\pi, \pi]$ and the network is trained until the sum of the squares of the errors becomes less than the error goal 0.001. The network is based on hidden neurons of logistic activations with biases and on a linear output neuron with bias. Comparative results are shown in table (2).

Table(2): Results of simulations for the function approximation problem

Algorithms	Min	Max	Mean	Tav	Succ
CBP	fail	--	--	--	0.0%
ABP	736	1986	1247.3	0.057405	64%
AIBP	227	1455	651.66	0.038281	88%

Form table (2), we conclude that the algorithm AIBP is the beast algorithm with respect to the succeeded simulations, number of epochs and the time.

4.3 Problem (3):(XOR Problem)

The last problem we have encountered 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 $E \leq 0.001$ within the limit of 1000 epochs, table (3) summarizes the result of all algorithms i.e for 50 simulations the minimum epochs for each algorithm is listed in the first column (Min), the maximum epoch for each algorithm is 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(3): Results of simulations for the XOR function

Algorithms	Min	Max	Mean	Tav	Succ
CBP	fail	--	--	--	0.0%
ABP	89	866	287.5	0.014633	64%
AIBP	29	831	147.35	0.0076288	74%

Form table (3), we conclude that the algorithm AIBP is the beast algorithm with respect to the succeeded simulations, number of epochs and the time.

Appendix

1- traingd: is matlab function (in the matlab toolbox) utilizes steepest descent direction with constant step-size to minimize error function E (training the network) known as standard Back propagation
2- traingda: is matlab function (in the matlab toolbox) utilizes steepest descent direction with adaptive step-size to minimize error function E (training the network) known as standard Adaptive Back propagation

References

1) Abbo K. and Zena T.(2011). "Minimization algorithm for training feed forward neural network", J. Of Educ and Sci. (to appear).
2) Andrei N. (2005). "Relaxed gradient descent method with backtracking for unconstrained Optimization". Technical Report No.c113.
3) Amir A., Mohammad B., and Abbas H. (2005). "Modified levenberg-marquardt method for NN training", World Academy of Science, Engineering and Technology 6. 46-48.
4) Clade B. and Michela R. (2007). "Generalizations of Aitken's process for accelerating the convergence of Sequences", J. of Computational and Applied Math. Vol (26). No 2, 171-189.
5) Daniel S., Vladimir. K and JiE P. (1997). "Introduction to multi-layer feed-forward neural networks.", Chemometrics and Intelligent Laboratory Systems 39 , 43-62.
6) Huajin T., Haizhou L. and Zhang Y. (2011). "Online learning and stimulus-driven responses of neuronsin visual cortex". Cogn Neurodyn 5. Springer:77–85.
7) Jarmo I., Jon K. and Jouni L. (2003). "Differential Evolution Training Algorithm for Feed-Forward Neural Networks", Neural Processing Letters 17: 93–105, Kluwer Academic Publishers. Printed in the Netherlands.
8) Johansson. E. Dowla F. and Goodman G. (1990). "Back propagation learning for Multi-Layer Feed –forward Neural", Networks using the Conjugate Gradient method Lawrence, ivermore National Laboratory. preprint UCRL-JC-104850.
9) Kostopoulos A. Sotiropoulos D. and Grapsa T. (2004). "A new efficient learning rate for Perry's spectral conjugate gradient Training method",1st International Conference ' From Scientific Computing to Computational Engineering'. 1st IC-SCCE.
10) Kuan G. and Hornik K. (1991). "Convergent of Learning algorithms with constant learning rates", IEEE Trans. Neural Networks, (2).

11) Livieris I. and Pintelas P. (2011) . "An improved spectral conjugate gradient neural network training algorithm", International J. on Artificial Intelligence Tools.
12) Livieris I., Sotiropoulos D., and Pintelas P. (2009). "On descent spectral CG algorithms for training recurrent neural networks", IEEE Computer Society. 13th Panellenic Conference of Informatics, pages 65–69.
13) Livieris I. and Pintelas P. (2008). "A survey on algorithms for training artificial neural networks", Technical Report No. TR08-01, Department of Mathematics University of Patras.
14) Mollar F. (1993). "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning", Neural Networks, 6. 525-533.
15) Nguyen D. and Widrow B. (1990). "Improving the learning speed of 2-layer neural network by choosing initial values of the adaptive weights", IEEE First International Jaint Conference on Neural Networks, (3).
16) Plagianakos. V., Sotiropouls D. and Vrahatis. M. (1998). "Automatic adaptation of Learning rate for Back-Propagation Neural networks", Recent advances in circuits and systems, Nikos E. Mastoraksi, ed, world Scientific.
17) Rumethart D. ,Hinton G. and Williams R (1986). "Learning Internal representations by Error propagation", In D. Rumelhart .J. Mc Clelland editors, Parallel Distributions Processing: Exploration in the Microstructure of Cognition. 318-362.
18) Rumelhart, D. E., Hinton, G. E. and Wiliams, R. (1986). "Learning representations by back-propagating errors", Nature, vol. 323, 533-536.
19) Sabeur A. and Farhat F. (2008). "Fast training of multi-layer perceptron with least mean fourth (LMF) Algorithm". International J. of soft computing. 3 (5), 359-367.
20) Shahla K., Ajaya D. and Jyothi N.(1997). "Application of artificial neural networks for the development of a signal monitoring system", Expert Systems, , Vol. 14, No. 2. 69-79.
21) Steven W. and Narciso . (1999). "Heuristic principles for the design of artificial neural networks", Information and Software Technology 41 (2), 109-119.
22) Zulhadi Z., Nor Ashidi M. and Shahrel A. (2010). "A Study on Neural Network Training Algorithm for Multi face Detection in Static Images". World Academy of Science, Engineering and Technology 62. 170-173.