IOP Conference Series: Materials Science and Engineering

PAPER • OPEN ACCESS

Fast Learning for Big Data Using Dynamic Function

To cite this article: T Alwajeeh *et al* 2017 *IOP Conf. Ser.: Mater. Sci. Eng.* **190** 012015

IOP Conf. Series: Materials Science and Engineering **190** (2017) 012015 doi:10.1088/1757-899X/190/1/012015

Fast Learning for Big Data Using Dynamic Function

T Alwajeeh[1], **A F Alharthi**[3], **R F Rahmat**[2], **R Budiarto**[3]

[1]Dept. of Computer Science & Engineering, College of CS&IT, Albaha University, Albaha P.O. Box 1988, Saudi Arabia
[2]Department of Information Technology, Faculty of Computer Science and Information Technology, University of Sumatera Utara, Medan, Indonesia
[3]Dept. of Computer Information System, College of CS&IT, Albaha University, Albaha P.O. Box 1988, Saudi Arabia

taa.2000@hotmail.com, afalharthi@bu.edu.sa, romi.fadillah@usu.ac.id, rahmat@bu.edu.sa

Abstract. This paper discusses an approach for fast learning in big data. The proposed approach combines momentum factor and training rate, where the momentum is a dynamic function of the training rate in order to avoid overshoot weight to speed up training time of the back propagation neural network engine. The two factors are adjusted dinamically to assure the fast convergence of the training process. Experiments on 2-bit XOR parity problem were conducted using Matlab and a sigmoid function. Experiments results show that the proposed approach signifcantly performs better compare to the standard back propagation neural network in terms of training time. Both, the maximum training time and the minimum training time are significantly faster than the standard algorithm at error threshold of 10^{-5}.

1. Introduction

Recently, we are entering the era of “big-data”, and as the development of high-speed signal processing, fast and efficient learning and signal representation is becoming an emergent research topic. Extreme learning machine (ELM) [1] is one of the leading trends for fast learning. Unlike the other traditional learning algorithms, for example, Back Propagation-based neural networks, or support vector machine (SVM)], the parameters of hidden layers of ELM are randomly established and need not be tuned, thus the training of hidden nodes can be established before the inputs are acquired.

Feedforward neural networks have been widely used in various areas of machine learning. Hidden nodes in a neural network architecture work as universal approximation provided that all the parameters of the networks are adjustable. The most representative training method for Artificial Neural Networks is back propagation (BP) algorithm. BP calculates the gradient of a loss function with respect to all the weights in the network and updates the weights for minimizing the loss function. Nevertheless, the parameter tuning of BP-based neural networks is usually time consuming and cannot handle the overfitting problem.

2. Related Works

Research on fast learning started from back-propagation algorithm, introduced by Werbos [2], and later popularized by Rumelhart et al. [3], which calculates the error function based on the weights of every data in the neural network, and then updates the weight with a new value, based on the activation function, in order to minimize the value of error function. However, back-propagation is known for the slow computing time, also inefficient for maintaining data in big size [4].

Researches have been done to improve learning time in neural network. Chandra and Sharma [5] introduces parameterized multilayer perceptron to process big data, with trigonometric functions.They also proposed parameterized deep neural network to reduce time usage of the learning process, by applying periodic function to parameterize the neural network weights [6]. Wong and Xu proposed hierarchical fast learning artificial neural network, which process data using feature compression based on canonical covariance [7]. Hinton and Teh [8] proposed a learning algorithm to improve learning speed of deep belief nets. Another strategy to improve learning time also proposed by Pasha [9] and Rahmat [10] by using distributed adaptive nested neural network.

3. Proposed Design

This paper uses standard back propagation artificial neural network (BP-ANN) as illustrated inf Figure 1. The input layer has i neurons. The hidden layer consists of four neurons with two weights, while the output layer has one neurons with one weights. A sigmoid function is used as activation function.

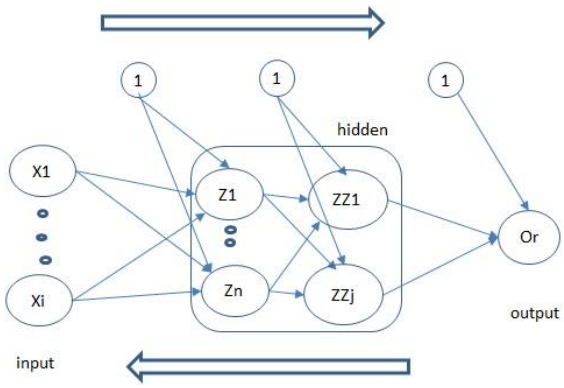

Figure 1. The BP-ANN model.

3.1. Momentum factor and traninig rate

Computing weights changes in neuron k of output layer and neuron j of hidden in BP-ANN involves two parameters; momentum factor (α) and training rate η. The larger value of η the faster the ANN converges. Usually, the value for η is randomly chosen by try and error from the value between 0 and 1. However, too big value of η may lead to oscillated training curve. The value of momentum factor α is also chosen randomly between 0 and 1. New weight adjustment Δw_{jk} at time t for each neuron j of hidden layer and neuron k of output layer is defined in equation (1).

$$\Delta w_{jk}(t+1) = w_{jk}(t) - \eta \frac{\partial E}{\partial w_{ik}(t)} + \alpha \Delta w_{jk}(t-1) \tag{1}$$

One important factor to speed up BP algorithm is the monotonicity of the error function during training for every epoch or iteration [11]. This paper uses an exponential function of error E for the dynamic training rate as defined in equation (2).

$$\eta_{DR}(E) = 1 + e^{(1+\sin(2E))} \tag{2}$$

Furthermore, in order to avoid overshoot weight in applying the training rate and momentum term as a dynamic function, a new approach is proposed by defining a relationship between the dynamic training rate and the dynamic momentum factor [12]. The momentum factor α_{DM} as an explicit function of training rate η_{DR} is defined in (3).

$$\alpha_{DM} = \sin(x + \varepsilon) + \sin\frac{1}{\eta_{DR}} \tag{3}$$

where $x = E \times f'(O_r)$, $f'(O_r)$ is the first derivation of activation function f which is defined as $f'(O_r) = O_r(1 - O_r)$ and ε is an absolute value, $0 < \varepsilon < 1$. We have investigated the best value of ε is 0.73. Substituting η_{DR} in equation (2) into equation (3) yield a new dynamic momentum factor as shown in equation (4).

$$\alpha_{DM} = [\ \sin(E(O_r\,(1 - O_r)) + \varepsilon) + \sin(\frac{1}{1 + e^{(1+\sin(2E))}})] \tag{4}$$

This approach maintains the weights are small as possible. Meanwhile, the momentum factor α and he training rate standard back propagation, SBP, manually train the training rate and momentum, where η and α are in the range of [0, 1].

3.2. The Proposed Training algorithm

The training algorithm of the proposed approach is given in Figure 2.

Step-1: Initialize randomly the weights
Step-2: Input: number of the neuron, hidden layer, the patterns, error threshold $E = 10^{-5}$
Step-3: While (*MSE>E*) do step 5-16
Step-4: For each training pair do step 6-15
Forward Propagation
Step-5: Compute input layer of hidden layer *Z*
Step-6: Compute input layer of hidden layer *ZZ*
Step-7: Compute input layer of output layer O_r and output value
Back Propagation
Step -8: Compute the error training
Step -9: Compute the error signal δ_r of the ANN
Step10: Compute the weight changes for each Δw_{jr} and bias Δw_{0r}
Step11: At zz_j compute the error signal δ_{-inj} and local gradient of error signal δ_j, using δ_r
Step12: Compute the weight correction for each Δv_{hj} and the bias Δv_{0j}
Step13: At z_h compute the error signal δ_{-inh} and local gradient of error signal δ_η using δ_r
Step14: At layer z_h compute the weight correction for each Δu_{ih}, and bias Δv_{0h}
Step15: Update the value for each output layer O_r, hidden layer zz_j and z_h
Step16: Compute the Mean Square Error: $MSE = 0.5 \times \frac{1}{P}\sum_{p=1}^{n}\sum_{k}^{i}(t_{kp} - o_{kp})$,
where P_i number of pattern for every epoch.

Figure 2. The training algorithm.

4. Experimental Results and Analysis

In this section we report the results obtained when experimenting our proposed method with 2-bits XOR Parity Problem. We use Matlab software running on Windows 8 machine with Intel Core i7 processor. The weights were generated randomly between [-0.4, 0.6] using equations (1-4). The obtained weights for *w1*, *w2*, *b1*, *b2* of hidden layer and *w3*, *b3* of output layer are:
w_1=[-0.3883 0.4175; -0.0872 0.2232]; w_2=[0.4575 -0.4463 ;0.1567 0.3496]; b_1= [0.2566; 0.2429]; b_2=[0.1101 0.1566] ; w_3=[0.4341 ;0.2878]; b_3= [- 0.3266].

The experiments were run 10 times for each value of error threshold and the average value is taken. The best results is shown in Table 1.

Table 1. The Best Result

Average Time (sec)	Average MSE	Average Epoch
0.4598	8.77E-06	269

Comparison to the standard BP training algorithm is depicted in Table 2. The value of α and η are varied between 0 and 1.

Table 2. The back progagation training algorithm results.

Time (sec)	MSE	Epoch	Value of η	α
588.6040	1.0000e-05	551105	0.1	0.1
279.0730	1.0000e-05	269082	0.2	0.2
154.2040	1.0000e-05	150827	0.3	0.3
97.0920	1.0000e-05	91009	0.4	0.4
63.9200	9.9999e-06	59658	0.5	0.5
15.1220	9.9998e-06	11911	0.6	0.6
3590	0.0657	1501828	0.7	0.7
3590	0.0661	3250451	0.8	0.8
57.0780	9.9999e-06	52445	0.9	0.4
33.9220	1.0000e-05	23760	1	0.5
27.6790	9.9998e-06	28389	0.8	0.5
56.6350	9.9998e-06	50429	0.8	0.4
32.0330	1.0000e-05	23765	1	0.5

The best time for the standard BP is 15.1220 seconds, which means that the proposed training algorithm is 30 times faster than the standard BP training algorithm.

Furthermore, the proposed training algorithm converges fast to the global minimum adn provides the best results at ε = 0.75 as shown in Figure 3.

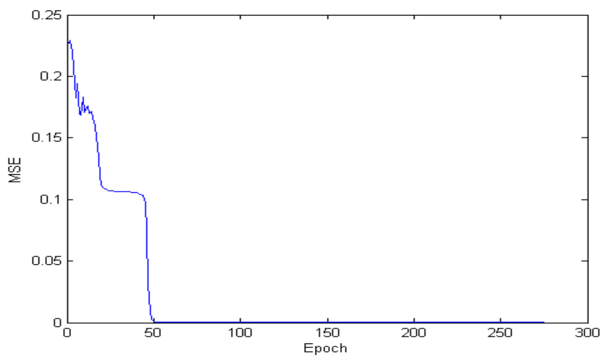

Figure 3. The convergence of the proposed training algorithm.

The proposed algorithm performs much better compare to the standard BP due the fact that the weights are automatically adjusted for every epoch in every layers through the dynamic dynamic momentum factor α_{DM} as defined in equation (4) as well as training rate η_{DR} as defined in equation (2). The use of implicit momentum function in the training rate in equation (3) makes the proposed algorithm able to control the growth of the weights. The dynamic momentum factor and training rate affect the weight for each hidden layer and output layer and eliminate the saturation training in the proposed algorithm. In addition, it uses an initial weight from interval [-0.4, 0.6] that narrows the search space compare to the standard Back Propagation.

5. Conclusion

An approach for fast learning on big data is proposed. The approach introduced a training algorithm to speed up the training time of Back Propagation neural network. The proposed algorithm considers a incorporates a dynamic function for auto-adjust two parameters: the momentum factor and the training rate. The proposed dynamic function eliminates the saturation of training time in Back Propagation artifical neural networks. In future, the proposed training algorithm can be applied for analyzing big data which needs extreme learning algorithm.

.

Acknowledgment

The author would like to thank Albaha University for the research funding under research contract number 1437/02.

References

[1] Deng C W, Huang G B, Xu J 2015 *Science China Information Sciences* **58** 020301(16), doi: 10.1007/s11432-014-5269-3.
[2] Werbos P 1974 *Beyond regression: new tools for prediction and analysis in the behavioral sciences*, (Cambridge MA: Harvard University Press).
[3] Rumelhart D E, Hinton G E and Williams R J 1986 *Nature* 323, pp. 533-536.
[4] Huang G B, Deng C W, Xu J and Tang J X 2015 *Science China Information Sciences* **58** 020301 pp 1-16.
[5] Chandra B and Sharma R K 2014 *Proc. of IEEE International Conference on Big Data (Anchorage, Alaska, USA)* pp 17-22.
[6] Chandra B and Sharma R K 2016 *Neurocomputing* 171 pp. 1205-1215.
[7] Wong L P and Xu J 2006 *Proc. of Int. Joint Conf. on Neural Networks (Montreal, Canada)* pp
[8] Hinton G E and Teh Y W 2006 *Neural Computation* 18 pp 1527-1554.
[9] Pasha M F, Rahmat R F, Budiarto R, and Syukur M 2010 *Int. Journal of Innovative Computing, Information and Control* 6 pp 1005-1022.
[10] Rahmat R F, Pasha M F, Syukur M, Budiarto R 2015 *Int. Arab Journal of Information Technology* **12** 6 pp 532-539.
[11] Li J, Lian L and Jiang Y 2010 *3rd Int. Conf. on Information and Computing (Wuxi, China)*, 4 pp 292–295.
[12] Shao H and Zheng G 2011 *Neurocomputing* **74** 5 pp 749–752.