Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 11 (2013) 32 – 39

Procedia Technology

The 4th International Conference of Electrical Engineering and Informatics (ICEEI 2013)

The Effect of Data Pre-Processing on Optimized Training of Artificial Neural Networks

Nazri Mohd Nawi[*], Walid Hasen Atomi, M. Z. Rehman

Faculty of Computer Science and Information Technology (FSKTM), University Tun Hussein Onn Malaysia (UTHM), 86400 Parit Raja, Batu Pahat, Johor, Malaysia.

Abstract

Recently, the popularity of artificial neural networks (ANN) is increasing since its capacity to model very complex problems in the area of Machine Learning, Data Mining and Pattern Recognition. Improving training efficacy of ANN based algorithm is a dynamic area of research and several papers have been reviewed in the literature. The performance of Multi-layer Perceptrons (MLP) trained with Back Propagation Artificial Neural Network (BP-ANN) method is highly influenced by the size of the datasets and the data-preprocessing techniques used. This work analyses the benefits of using pre-processing datasets using different techniques in-order to improve the ANN convergence. Specifically Min-Max, Z-Score and Decimal Scaling Normalization preprocessing techniques were evaluated. The simulation results show that the computational efficiency of ANN training process is highly enhanced when coupled with different preprocessing techniques.

Keywords: Artificial neural networks; back propagation; gradient descent; gain value; pre-processing data.

1. Introduction

The Artificial Neural Networks (ANN) has become popular recently and is one of the most effective computational intelligence techniques applied in Pattern Recognition Data Mining and Machine Learning. The main power of ANN lie in its ability to estimate multifaceted and non-linear relationships between input and output data

* Corresponding author. Tel.: +601-07-4533603
E-mail address: nazri@uthm.edu.my

by learning from an instance [1]. Multi-layer Perceptrons (MLP) are the most common type of ANN. The general structure of MLP comprises of three layers (i.e. Input layer, hidden layer and output layer). Back Propagation Artificial Neural Network (BP-ANN) is the most common method used to train MLP [2]. This learning algorithm is a gradient-descent based algorithm which suffers from some drawbacks such as getting trapped in local minima, slow convergence and network stagnancy [3-5].

Improving training ability of ANN based algorithm is a dynamic area of research and many papers have been published until today on this very topic. Early research on back propagation (BP) algorithms saw enhancements on: (i) selection of better error functions [6-13]; (ii) different choices for activation functions [8, 14] and, (iii) selection of dynamic learning rate and momentum [15-17].

Bishop [18] suggested the use of many optimization techniques in his book for improving efficiency of the error minimization process. Among these are methods of Fletcher and Powel [19] and the Fletcher-Reeves [20] that improve the conjugate gradient method of Hestenes and Stiefel [21] and different variants of Quasi-Newton algorithms by Huang [22].

Later, it was found out that the techniques for Pre-processing the data also plays an important role in effecting the performance of MLP, where pre-processing data encourage the high accuracy and less computational cost associated to the learning phase.

In this paper, we analyze and explore the performance of several MLP models on three pre-processing methods in order to modify and reduce the training-set size, removing atypical or noisy training samples and correcting possible erroneous identifications of the training samples. The main goals are to decrease the computational cost and to accelerate the learning process.

Four ANN models are presented in this paper; a simple modification to the gradient based search direction is used by all these methods as summarized by Bishop [23]. Furthermore, the gradient based search direction is locally altered by a gain value used in the activation function of the corresponding node to improve the convergence rates. In the simulations, we employed three benchmark datasets obtained from the University of California Irvine (UCI), Machine Learning Repository. The datasets used are Wine recognition, Haberman's survival and Iris.

The remaining paper is organized as follows: Section 2 and 3 presents the theoretical background of the modification to the gradient based, the implementation in other optimization techniques and pre-processing techniques. The section 4 provides the results and discussions. Finally, the conclusion is settled in the Section 5.

2. The gradient descent modification

Nowadays, the Multilayer Perceptrons (MLP) trained with the back propagation (BP) is one of the most common method used for classification purpose [24]. This method has the capacity of organizing the representation of the data in the hidden layers with high power of generalization. Normally, its architecture has three consecutive layers: input, hidden and output layer [25-27]. The MLP with one layer can build a linear hyper plane, with two layers it can build convex hyper plane and with three it can build any hyper plane.

The MLP training using gradient descent back propagation requires careful selection of parameters such as; network topology, early weights and biases, learning rate value, activation function, and the value for gain in the activation function. An improper choice of these parameters can lead to slow network convergence or network stagnancy [3-4].

Seeing these problems, previous researchers have proposed many variations in gradient descent algorithm. Lately, Nazri [28] proposed a modification by changing the gradient based search direction using a gain value. For the comparison purposes, this study will apply the gradient descent with gain (GDG) algorithm proposed by Nazri [28] due to its successful implementation and proven not to suffer the local minimum as compared to the traditional gradient descent. The GDG algorithm proposed by Nazri [28] for changing the gradient based search direction using a gain value is as follows:

Initialize the initial weight vector with random values and the vector of gain values with unit values. Repeat the following Steps 1 and 2 on an epoch-by-epoch basis until the given error minimization criteria are satisfied.

Step 1: By introducing gain value into activation function, calculate the gradient of error with respect to weights by using Equation (5), and gradient of error with respect to the gain parameter by using Equation (7)

Step 2: Use the gradient weight vector and gradient of gain vector calculated in step 1 to calculate the new weight vector and vector of new gain values for use in the next epoch.

2.1. Derivation of the expression to calculate gain value

Consider a MLP network, as used in standard back propagation algorithm[18]. Suppose that for a particular input pattern o^0, the desired output is the teacher pattern $t = [t_1..t_n]^T$, and the actual output is o_k^L, where L is the output layer. The error function on that pattern is defined as;

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

Let o_k^s be the activation values for the k^{th} node of layer s, and let $o^s = [o_1^s ... o_n^s]^T$ be the column vector of activation values in the layer s and the input layer as layer 0. Let w_{ij}^s be the weight values for the connecting link between the i^{th} node in layer $s-1$ and the j^{th} node in layer s, and let $w_j^s = [w_{1j}^s ... w_{nj}^s]^T$ be the column vector of weights from layer $s-1$ to the j^{th} node of layer s. The net input to the j^{th} node of layer s is defined as $net_j^s = (w_j^s, o^{s-1}) = \sum_k w_{j,k}^s o_k^{s-1}$, and let $net^s = [net_1^s ... net_n^s]^T$ be the column vector of the net input values in layer s. The activation value for a node is given by a function of its net inputs and the gain parameter c_j^s;

$$o_j^s = f(c_j^s net_j^s) \tag{2}$$

where, f is any function with bounded derivative . This information is now used to derive an expression for modifying gain values for the next epoch. Most of gradient based optimization methods use the following gradient descent rule:

$$\Delta w_{ij}^{(n)} = -\eta^{(n)} \frac{\partial E}{\partial w_{ij}^{(n)}} \tag{3}$$

where $\eta^{(n)}$ is the learning rate value at step n and the gradient based search direction at step n is

$$d^{(n)} = -\frac{\partial E}{\partial w_{ij}^{(n)}} = g^{(n)}$$

In the proposed method the gradient based search direction is modified by including the variation of gain value to yield;

$$d^{(n)} = -\frac{\partial E}{\partial w_{ij}^{(n)}}(c_j^{(n)}) = g^{(n)}(c_j^{(n)}) \tag{4}$$

The derivation of the procedure for calculating the gain value is based on the gradient descent algorithm. The error function as defined in Equation (1) is differentiated with respect to the weight value w_{ij}^s. The chain rule yields;

$$\frac{\partial E}{\partial w_{ij}^s} = \frac{\partial E}{\partial net^{s+1}} \cdot \frac{\partial net^{s+1}}{\partial o_j^s} \cdot \frac{\partial o_j^s}{\partial net_j^s} \cdot \frac{\partial net_j^s}{\partial w_{ij}^s} = [-\delta_1^{s+1} ... -\delta_n^{s+1}] . \begin{bmatrix} w_{1j}^{s+1} \\ \vdots \\ w_{nj}^{s+1} \end{bmatrix} . f'(c_j^s net_j^s) c_j^s . o_j^{s-1} \tag{5}$$

where, $\delta_j^s = -\frac{\partial E}{\partial net_j^s}$. In particular, the first three factors of Equation (5) indicate that the following equation holds;

$$\delta_1^s = (\sum_k \delta_k^{s+1} w_{k,j}^{s+1}) f'(c_j^s net_j^s) c_j^s \tag{6}$$

It should be noted that, the formula as described in Equation (6) to calculate δ_1^s is the same as used in the standard back propagation algorithm [22] except for the presence of the gain value in the expression. The learning rule for calculating weight values as given in Equation (3) is derived by combining Equation (5) and (6).

This way, the gradient error with respect to the gain parameter can also be calculated by using the chain rule as previously described; it is easy to compute as;

$$\frac{\partial E}{\partial c_j^s} = (\sum_k \delta_k^{s+1} w_{k,j}^{s+1}) f'(c_j^s net_j^s) net_j^s \tag{7}$$

Then, the gradient descent rule for the gain value becomes;

$$\Delta c_j^s = \eta \delta_j^s \frac{net_j^s}{c_j^s} \tag{8}$$

At the end of every epoch the new gain value is updated using a simple gradient based method as;

$$c_j^{new} = c_j^{old} + \Delta c_j^s \tag{9}$$

2.2. The implementation on Gradient Descent Method with Adaptive Gain Algorithm (GD/AG)

In gradient descent method, the search direction at each step is given by the local negative gradient of the error function, and the step size is determined by a learning rate parameter. Suppose at step n in gradient descent algorithm, the current weight vector is w^n, and a particular gradient based search direction is d^n. The weight vector at step *n+1* is computed by the following expression:

$$w^{(n+1)} = w^n + \eta^n d^n \tag{10}$$

where, η^n is the learning rate value at step n.
By using the proposed method, the gradient based search direction is calculated at each step by using Equation (4).

The complete GD/AG algorithm works as follows;

Step 1 Initialize the weight vectors randomly, the gradient vector g_0 to zero and gain vector to unit values. Let the first search direction d_0 be g_0. Set $\beta_0 = 0$, $epoch = 1$ and $n = 1$. Let Nt be the total number of weight values. Select a convergence tolerance value as CT.

Step 2 At step n, evaluate the gradient vector $g_n(c_n)$.

Step 3 Evaluate $E(w_n)$. If $E(w_n) < CT$ then STOP training ELSE go to step 4.

Step 4 Calculate a new gradient based search direction which is a function of gain parameter: $d_n = -g_n(c_n) + \beta_{n-1} d_{n-1}$

Step 5 If $n > 1$ THEN,

Step 6 Update w_n : $w_{n+1} = w_n - \eta_n^* d_n$

Step 7 Evaluate the new gradient vector $g_{n+1}(c_{n+1})$ with respect to gain value c_{n+1}.

Step 8 Calculate the new gradient based search direction: $d_{n+1} = -g_{n+1}(c_{n+1}) + \beta_n(c_n)d_n$

Step 9 Set $n = n + 1$ and go to step 2.

3. Data pre-processing techniques

Data pre-processing is an important step in the data mining process. Mostly, data gathering methods are lightly controlled, resulting in outliers, impossible data combinations, and missing values, etc. Analyzing data that has not been carefully separated for such problems can produce confusing results. Thus, the depiction and quality of data is first and foremost before running any analysis [29]. The quality, reliability and availability are some of the factors that may lead to a successful data interpretation by a neural network. If there is inappropriate information present or noisy and unreliable data, then knowledge discovery becomes very difficult during the training process. Data preparation and filtering steps can take considerable amount of processing time [30] but once pre-processing is done the data become more reliable and robust results are achieved. This study will compare three methods for data normalization including min-max normalization, z-score normalization and normalization by decimal scaling.

4. Results and discussions

The main focus of this study is to improve the accuracy of ANN models by using three selected pre-processing techniques. Before discussing the simulation test results there are certain things that need to be explained such as tools and technologies, network topologies, parameters settings, data sets and training methodology used during the experimentation process. The discussion is as follows:

The workstation used for carrying-out the experimentation comes with a 2.33 GHz Core-2 Duo Processor, 1 GB of RAM while Microsoft XP (Service Pack 3) is used as the operating system. Three gradient descent algorithms proposed by Nazri [28] as described in Section 3 and one traditional gradient descent algorithm were used to carry out simulations on MATLAB 7.

Three layer back propagation neural networks is used while the hidden layer is kept fixed to 5 nodes. For the first trials, the global learning rate of 0.4, momentum value of 0.3 and gain value of 0.3 are varied adaptively between the range of [0, 1] randomly and were selected for the entire training and testing. While log sigmoid activation function is used as the transfers function from input layer to hidden layer and from hidden layer to output layer.

For each data set, each trial is limited to 5000 epochs. A total of 15 trials are run for each parameter to validate the best possible results. The network results are stored in the result file for each trial. Mean Square Error (MSE) is used to verify the accuracy in the results.

On each problem, the following four algorithms were analyzed and simulated, here two algorithms are proposed by Nazri [25][28] and the other two are the standard algorithms:

1. Traditional Gradient Descent with Momentum (GDM).
2. Gradient Descent with Line search (GDL)
3. Gradient Descent with Gain (GDG).
4. Gradient Descent with Gain and Line search (GDGL).

Three pre-processing techniques used are as follows:

1. PMMN1: Pre-processing Technique (Min-Max Normalization).
2. PZSN2: Pre-processing Technique (Z-Score Normalization).
3. PDSN3: Pre-processing Technique (Decimal Scaling Normalization).

4.1. Benchmark datasets

The simulations were carried out on three datasets taken from UCI Machine Learning Repository. For each dataset, we have estimated the overall accuracy by 5-fold cross-validation: each data set was divided into five equal parts, using four folds as the training set and the remaining block as independent test set. Those three datasets are the Wine recognition, Haberman's survival and the Iris.

4.2. Classification results

Using the prepared datasets, set of experiments were performed. The data proportion applied is as stated below; data transformation such as normalization improved the accuracy and efficiency of ANN models.

Table 1. Classification of Haberman's Survival

HABERMAN DATASET

Pre-Processing	**Accuracy**			
	GDM	**GDG**	**GDGL**	**GDL**
PMMN1	95.55%	97.61%	96.71%	96.66%
PZSN2	95.67%	97.17%	95.92%	95.88%
PDSN3	95.45%	97.01%	97.19%	97.11%

Table 2. Classification results for Wine Recognition

WINE RECOGNITION DATASET

Pre-Processing	**Accuracy**			
	GDM	**GDG**	**GDGL**	**GDL**
PMMN1	95.11%	96.90%	98.99%	94.32%
PZSN2	95.08%	97.73%	96.77%	95.44%
PDSN3	96.23%	96.64%	97.66%	95.88%

Table 3. Classification results for IRIS

IRIS DATASET

Pre-Processing	**Accuracy**			
	GDM	**GDG**	**GDGL**	**GDL**
PMMN1	98.41%	99.87%	99.12%	98.01%
PZSN2	96.06%	99.46%	98.73%	98.76%
PDSN3	97.20%	94.47%	96.12%	95.89%

Table 1, Table 2 and Table 3, show the classification results obtained with several pre-processing techniques (PMMN1, PZSN2 and PDSN3) using GDM, GDG, GDGL and GDL). The values marked with bold typeface indicate the best results.

From these results, some initial remarks can be drawn. Firstly, for the majority of the datasets, there exists at least one pre-processing technique whose classification accuracy is better as compared to others. Also, we detect that after applying the pre-processing techniques with several MLP-ANN methods described previously, the PMMN1 technique was the best one preserving the overall accuracy as much with the GDG as well as GDGL models.

On the other hand, when comparing the overall classification accuracy of the gradient descent models, we found that the GDG model generally has a promising behavior for all three datasets and three pre-processing techniques with an accuracy of 99.87%. In other words, the accuracy obtained with GDGL can differ from one problem to another depending on the dataset used, yet it outperforms the traditional GDM except for IRIS dataset with PDSN3. On the other hand, it was observed that with the addition of pre-process techniques, the GDG model behavior is effected more than the other models especially when PMMN1 technique is applied.

The IRIS dataset establishes a very certain case, in which GDG model behavior is strongly affected. In all cases, the results obtained with PMMN1 pre-processing technique is always higher than the overall accuracies obtained with any other ANN model. A similar situation is observed with Haberman's Survival data set when PMMN1 is used together with GDG model, it still outperform others models with the highest accuracy of 97.61%.

5. Conclusions

Nowadays, in the ANN, the high computational cost connected with the learning process of the MLP is a grave problem. This cost is directly linked with the training dataset size. In this study, we proposed to reduce the computational cost of ANN training by introducing pre-processing techniques (such as; Min-Max, Z-Score and Decimal Scaling Normalization). For that, four variations of well-known gradient descent methods were used. The simulations results show that the use of pre-processing techniques increased the accuracy of the ANN classifier by at least more than 95%. This achievement could be interpreted in an important computational cost diminution related to the learning process of the network. On the other hand, it was also observed that in many datasets the classifier behavior was increased and only a few cases show loss in the classifier effectiveness. Finally, the methodology proposed in this work can be advantageous when the computational cost linked to the ANN learning is expected. The results show that the pre-processing techniques increase the robustness of the proposed algorithm [25] [28] and increase the training efficacy of MLP's.

Acknowledgments

The Authors would like to thank Office of Research, Innovation, Commercialization and Consultancy Office (ORICC), Universiti Tun Hussein Onn Malaysia (UTHM) and Ministry of Higher Education (MOHE) Malaysia for financially supporting this Research under Fundamental Research Grant Scheme (FRGS) vote no. 1236.

References

[1] Azadeh, M. Sheikhalishahi, M. Tabesh, A. Negahban, The Effects of Pre-Processing Methods on Forecasting Improvement of Artificial Neural Networks, Australian Journal of Basic and Applied Sciences, 5(6): 570-580, 2011.

[2] Thimm G., Moerland F., Fiesler E. The Interchangeability of Learning Rate and Gain in Backpropagation Neural Networks. Neural Computation, vol. 8(2); 1996. p. 451- 460

[3] M.Z. Rehman, Nazri Mohd Nawi, The Effect of Adaptive Momentum in Improving the Accuracy of Gradient Descent Back Propagation Algorithm on Classification Problems. CCIS, Vol. 179, Issue No. 6, Springer Heidelberg. 2011.

[4] M.Z. Rehman, Nazri Mohd Nawi, and M. I . Ghazali, Predicting Noise-Induced Hearing Loss (NIHL) and Hearing Deterioration Index (HDI) in Malaysian Industrial Workers using GDAM Algorithm. Journal of Engineering and Technology, Vol. 3, Issue No. 1. 2012.

[5] Nazri Mohd Nawi, M.Z. Rehman, and M. I . Ghazali, Noise-Induced Hearing Loss Prediction in Malaysian Industrial Workers using Gradient Descent with Adaptive Momentum Algorithm. International Review on Computers and Softwares (IRECOS), Vol. 6, Issue No. 5. 2011.

[6] A. Van Ooyen and B. Nienhuis, Improving the convergence of the back-propagation algorithm. Neural Networks; 1992. 5: p. 465-471.

[7] M. Ahmad and F.M.A. Salam, Supervised learning using the cauchy energy function. International Conference on Fuzzy Logic and Neural Networks; 1992.

[8] Pravin Chandra and Yogesh Singh, An activation function adapting training algorithm for sigmoidal feedforward networks. Neurocomputing, 2004. 61: p. 429-437.

[9] Krzyzak A., Dai W., and Suen C. Y., Classification of large set of handwritten characters using modified back propagation model. Proceedings of the International Joint Conference on Neural Networks, 1990. 3: p. 225-232.

[10] Sang Hoon Oh, Improving the Error Backpropagation Algorithm with a Modified Error Function. IEEE TRANSACTIONS ON NEURAL NETWORKS, 1997. 8(3): p. 799-803.

[11] Hahn-Ming Lee, Tzong-Ching Huang, and Chih-Ming Chen, Learning Efficiency Improvement of Back Propagation Algorithm by Error Saturation Prevention Method. IJCNN '99, 1999. 3: p. 1737-1742.

[12] Sang-Hoon Oh and Youngjik Lee, A Modified Error Function to Improve the Error Back-Propagation Algorithm for Multi-Layer Perceptrons. ETRI Journal, 1995. 17(1): p. 11-22.

[13] S. M. Shamsuddin, M. Darus, and M. N. Sulaiman, Classification of Reduction Invariants with Improved Back Ppropagation. IJMMS, 2002. 30(4): p. 239-247.

[14] S. C. Ng, et al., Fast convergence for back propagation network with magnified gradient function. Proceedings of the International Joint Conference on Neural Networks 2003, 2003. 3: p. 1903-1908.
[15] R.A. Jacobs, Increased rates of convergence through learning rate adaptation. Neural Networks, 1988. 1: p. 295–307.
[16] M.K. Weir, A method for self-determination of adaptive learning rates in back propagation. Neural Networks, 1991. 4: p. 371-379.
[17] X.H. Yu, G.A. Chen, and S.X. Cheng, Acceleration of backpropagation learning using optimized learning rate and momentum. Electronics Letters, 1993. 29(14): p. 1288-1289.
[18] Bishop C. M., Neural Networks for Pattern Recognition.: Oxford University Press; 1995
[19] R. Fletcher and M. J. D. Powell, A rapidly convergent descent method for nlinimization. British Computer J., 1963: p. 163-168.
[20] Fletcher R. and Reeves R. M., Function minimization by conjugate gradients. Comput. J., 1964. 7(2): p. 149-160.
[21] M. R. Hestenes and E. Stiefel, Methods of conjugate gradients for solving linear systerns. J. Research NBS, 1952. 49: p. 409.
[22] HUANG H.Y., A unified approach to quadratically convergent algorithms for function minimization. J. Optim. Theory Appl., 1970. 5: p. 405-423.
[23] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, Learning internal representations by error propagation. in D.E. Rumelhart and J.L. McClelland (eds), Parallel Distributed Processing, 1986. 1: p. 318-362.
[24] Haykin S. Neural Networks, A Comprehensive Foundation. 2nd ed. New Jersey: Prentice Hall; 1999.
[25] Cristianini N., J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, Cambridge UK, 2000.
[26] Dasarathy B. V., minimal consistent set (MCS) identification for optimal nearest neighbour decision system design, Transaction on System Man and Cybernatics. 24(3). 1994.
[27] Shertinsky A., Rosalind P., On the efficiency of the orthogonal Least Squares Training Method for Radial Function Networks, IEEE Transaction on Neural Networks. 7(1). 1996.
[28] Nazri Mohd Nawi, Rozajda Ghazali and Mohd Najib Mohd Salleh, The development of improved back-propagation neural networks algorithm for predicting patients with heart disease. LNCS, Vol. 6377, Issue No. M4D; 2010.p.. 317-324.
[29] D. Pyle, Data Preparation for Data Mining. 1999. Los Altos, California: Morgan Kaufmann Publishers.
[30] S. Kotsiantis, D. Kanellopoulos, & P. Pintelas, Data Preprocessing for Supervised Leaning. International Journal of Computer Science , Vol. 1, Issue No. 2, 2006; p. 111-117.