Open access Journal International Journal of Emerging Trends in Science and Technology
IC Value: 76.89 (Index Copernicus) Impact Factor: 4.219 **DOI:** https://dx.doi.org/10.18535/ijetst/v5i1.02

Improvisation of Training Algorithm through Hybridization for Rule Extraction

Authors
Vinita Srivastava[1], Chitra Dhawale[2]
[1]Department of I.I.C.C. , R.T.M. Nagpur University, Nagpur India
Email: *vinni.sara12@gmail.com*
[2]Department of Computer Science, Amravati
Email: *cadhawale@rediffmail.com*

Absrtact
The Artificial Neural Network is widely used for classification. In classification, training and learning of the network in which weights and biases of the network neuron are computed to give expected output, is a complex task. In this paper, we propose hybridization of back propagation and LevenbergMarquardt training algorithms. The gradient derivative with respect to the weight of the network of the gradient descent algorithm is used in augmenting Hessian matrix of Levenberg Marquardt training algorithm to update the weight and bias of the network to converge to output. The hybrid algorithm is experimented on two data sets. Experimental results show that it helps to achieve better network performance and extracts fewer rules.
Keywords: *Rule Extraction, training algorithm, back propagation, Levenberg-Marquardt, network performance.*

Introduction

The main process in an Artificial Neural Network is a supervised learning of the network in which network learns the pattern of expected output for a given input by computing weights and bias of the network neurons. This is mainly achieved by the training algorithm of the network. Artificial Neural Network learns the pattern of the expected output of a given input through training algorithm. Training algorithm trains the network by computing weights and bias of the network neurons. The speed of feed forward neural networks is slower than required for its applications. The use of slow gradient-based learning algorithms and iteratively tuning of parameters of the networks by these training algorithm are the main reasons of slow speed of feed forward neural networks [2]. Gradient descent-based methods are mainly used in a feed forward neural network, which takes considerable time and readily converge to local minima[2]. Back propagation technique is the most popular gradient descent training algorithm. It reaches the minimum of quadratic surfaces using in Newton's method. The computational complexity increases by calculating inverse Hessian matrix. Researchers have worked on improving Levenberg Marquardt (LM) based back convergence of Levenberg Marquardt[8].[9] Has proposed an Back Propagation trained with a Cuckoo search algorithm for superior convergence speed of the hybrid neural networks learning method.

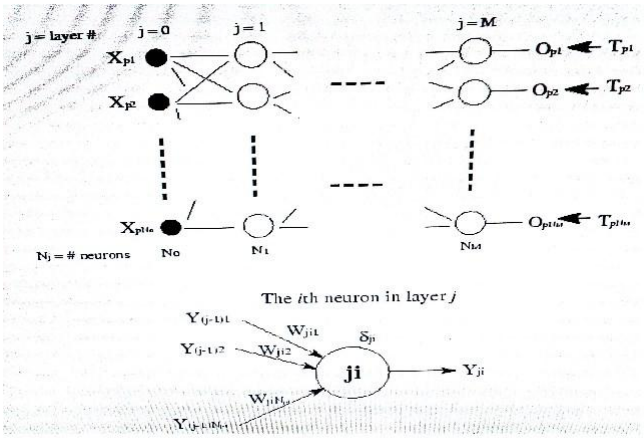

Fig. 1. Back Propagation Neural Network

Back Propagation

The Back propagation algorithm (BP) is a gradient descent training algorithm. It calculates the error at the output and the gradient of this error. Weights and biases of the ANN are adjusted in descending gradient direction accordingly. It can be expressed as expression $w = rE$, where the parameter $\eta > 0$ is the learning rate that controls the learning speed [7]. The back propagation algorithm is used as a training algorithm for classification application of ANN. The error value of response output and expected output is calculated. The connection weights and biases are adjusted based on the value of error. The input patterns are presented repeatedly to the network till the error value is minimized [11].

Refer to the Fig 1 [11] that illustrates the back propagation multilayer network. The network has M layers which has input of pth instance of training data sample. Each layer has neurons represented by N. The output represented by T is at the output layer [11].

Levenberg-Marquardt

The Levenberg-Marquardt algorithm (LM) is an approximation to the Newton method used also for training ANNs.

Newton's method for optimizing a performance index F(x) is given by

$$x_{k+1} = x_k - A_k^{-1} g_k \quad (1)$$

where $A_k = \nabla^2 F(x)_{x=x}k$ and $g_k = \nabla F(x)_{x=x}k$ are the hessian and the gradient of F(x), respectively where xk is net parameters at time k. F(x) is the sum of squares e(x) over the N targets.

$$F(x) = \sum_{i=1}^{N} e2i(x) = e^T(x)\, e(x) \quad (2)$$

Then the gradient would be

$$\nabla F(x) = 2J^{T(x)}\, e(x) \quad (3)$$

Where $J^{T(x)}$ is the Jacoban matrix formed by $\frac{\partial e^i(x)}{\partial x_j}$

LM, based on Newton method, calculates the error of the network with second order expression whereas BP algorithm calculates with first order expression[3]. LM updates the ANN weights as follows:[3]

$$\Delta w = -\left[\mu I + \sum_{p=1}^{P} J^P(w)^T J^P(w)\right]^{-1} \bigtriangledown E(w) \quad (4)$$

Where Jp(w) is the Jacobian matrix of the error vector ep(w) evaluated in w,and I is the identity matrix. The vector error ep(w) is the error of the network for pattern p, that is, ep(w) = tp - op(w). The parameter μ is decreased in case of reduction in error, by dividing it by a factor β. In the same way it is increased by multiplying by a factor β in other case. The network output, the error vectors, and the Jacobian matrix for each pattern is calculated and also δw using (4) and again assesses the error with w + δw as network weights. If the error has reduced , μ is divided by β, the new weights are maintained, and the process repeats again The process starts again if μ is decreased otherwise δw is calculated [3].

Hybrid Algorithm

The network is prepared for training when its weights and biases are initialized.The network performance is augmented by tuning the values of weights and biases.The default performance function for feed forward networks is mean square error. The training of the network can be done in either incremental mode or batch mode.

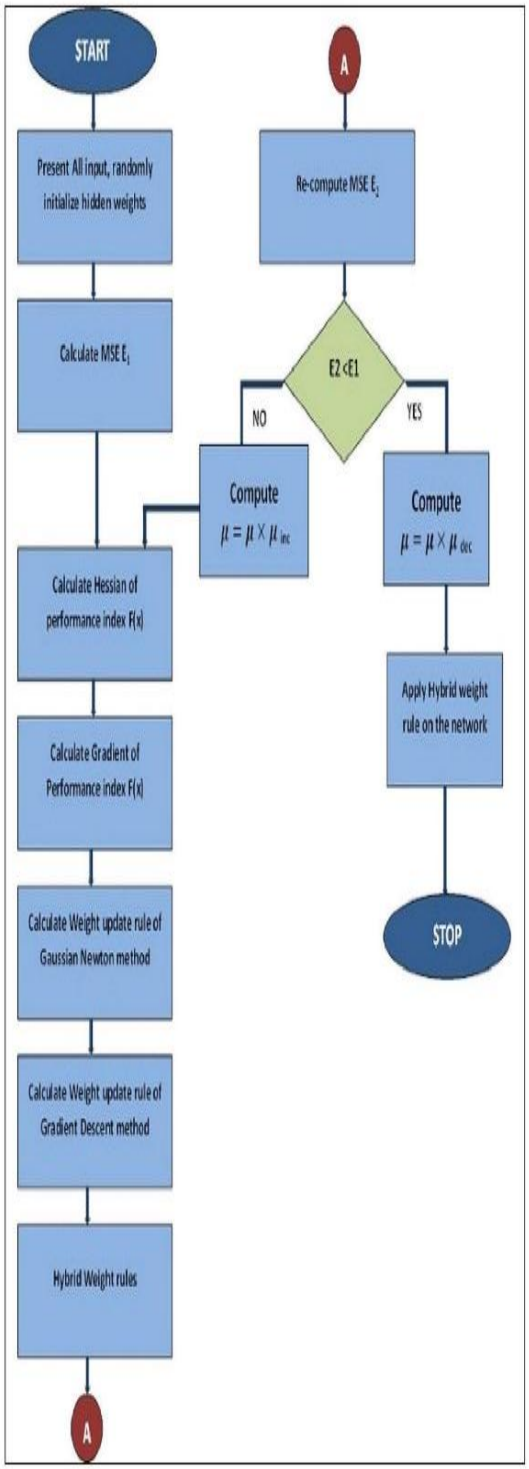

Fig 2. Flowchart of Algorithm

In incremental mode, after estimation of gradient the weights are updated after each input. On the contrary, all the inputs in the training set are applied to the network before the weights are updated. Gradient descent apprises the network weights and biases in the direction in which the performance function reduces swiftly. The algorithm can be written as xk+1 = xk- (lr*gx) for an iteration, where xk is a vector of current weights and biases, gx is the current gradient, and lr is the learning rate. This equation is iterated and the progress is continously updated until the network converges. During training,. The performance, the magnitude of the gradient of performance and the number of validation checks are most crucial. The training is stoped on the basis of magnitude of the gradient and the number of validation checks. As the training reaches a lowest of the performance, the gradient becomes very small. the training will be stoped if the magnitude of the gradient is less than 1e- 5. The performance function estimates the performance of the network and modifies the weights and biases of the network to change to smaller weights and biases, and this forces the network response to be smoother and less likely to overfit. The network compares the calculated performance with the best performance of the network and updates the best performance value with the current lower performance value. The small change in neuron weights will result in a change in the output of that layer, which again will be the input of the next layer. This way that changed values will affect the output of the output layer in such a way that mean squared error E1 will be minimized.

In the proposed hybridization of training algorithm, the gradient derivative with respect to the weight of the network of the gradient descent algorithm is augmented with the one calculated by Hessian matrix. The Mean Squared Error E2 is calculated with resulted weights. The weight and bias of the network are updated with E2 if it is less than E1. As shown in Fig 2, mu (μ) is calculated. Mu is the control parameter for the algorithm used to train the neural network. Choice of mu directly affects the error convergence. , With reduction in performance function the mu is decreased and is increased with increase the performance function. After every iteration the value of performance function is reduced. The parameter mu is the initial value for μ. This value is multiplied by mu_{dec} on reduction of performance function. It is multiplied by mu_{inc} on

increase of the performance function. If mu becomes larger than mu_{max},the algorithm is paused.The performance of the network is augmented.

Experimental Results

To evaluate the performance of a hybrid training algorithm, we experimented the method with two data sets on three layered feed forward network. The ILPD (Indian Liver patient Data set) is used to explain the effect of the proposed training algorithm. The ILPD data set has 583 instances of patients. Each patient is characterized by 10 continuous inputs. The task is to classify these patients into one of the two classes. The other data set is Pima Indians Diabetes data. The PID data set consists of 768 instances of patients. Each patient is characterized by 8 continuous inputs. The problem is to classify these patients into one of the two classes. The Experimental results are two folds. Firstly, the results show the best performance value attained by the network in an epoch number. Secondly the effect of hybrid training algorithm on number of rules extracted from the network with the accuracy of the rule.

Table I. Dataset

Data set	No of instance	No of attribute	No of Classes
PID	768	8	2
ILPD	583	10	2

Bestepoch indicates the iteration at which the validation performance reached a minimum. Fig 3 and Fig 4 shows results of best epoch and best performance respectively, for ILPD and PID data sets from three layered feed forward neural network using regular Levenberg Marquardt training algorithm and using a new hybrid training algorithm.

The next fold of experiment has been done to know the effect of the training algorithm for rule extraction from the neural network. Previous work[1] has been done on this aspect of training algorithm. Fig 5 and Fig 6 shows the number of rules extracted and the accuracy of rules respectively, for ILPD and PID data sets from three layered feed forward neural network using regular Levenberg Marquardt training algorithm and using a new hybrid training algorithm.

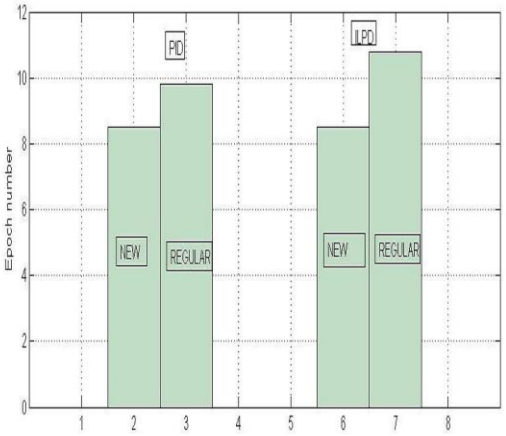

Figure 3. Epoch number of best performance Value

Sets from three layered feed forward neural network using regular Levenberg Marquardt training algorithm and using a new hybrid training algorithm.

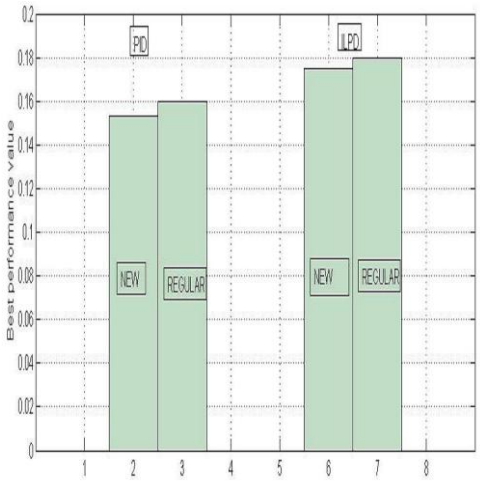

Figure 4. Best Performance value.

Discussion

In Fig 3 the value of Best-epoch of the network trained with new hybrid training algorithm is less than the value of the network trained with Levenberg Marquardt training algorithm. This shows that network reaches best performance value in less number of epochs if trained with hybrid training algorithm. In other words we can say that network trains faster in comparison to the training

with regular training algorithm. Fig 4 shows the value of best performance of the network trained with hybrid training algorithm and with Levenberg Marquardt training algorithm. The value of best performance is the attained minimum value of the difference of target value and output value. Result shows that the above value is less for the network trained with hybrid training algorithm hence network converge better towards target values. Experimental results also show that it gives fewer rules with higher classification accuracy if trained with hybrid training algorithm.

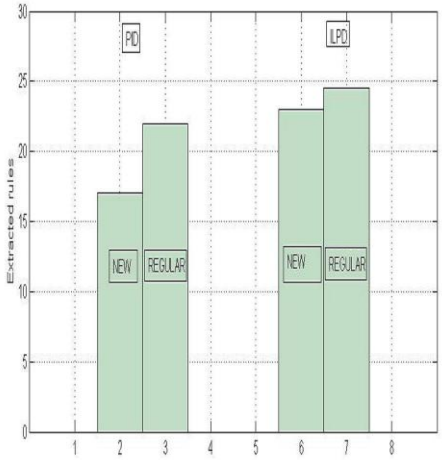

Figure. 5. Number of Extracted Rules.

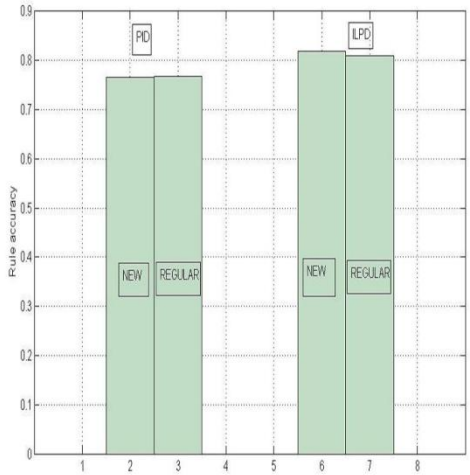

Figure. 6. Classification Accuracy.

Conclusion

In this paper, The Proposed algorithm is experimented only on two data set. The proposed algorithm gives better network performance in comparison to traditional Levenberg Marquardt. The network trained with proposed hybrid training algorithm converges more and faster in comparison to the network trained by traditional Levenberg Marquardt training algorithm. The hybrid algorithm is experimented on datasets with continuous inputs only. Hence the future scope of the work done is to experiment the proposed algorithm for discrete attributes also and to compare accuracy of the hybrid training algorithm with resilient back-propagation algorithm which gives better accuracy than Levenberg Marquardt algorithm[10].

References

1. B.Sharma, K.Venugopal Comparison of Neural Network Training Functions for Hematoma Classification in Brain CT Images, IOSR Journal of Computer Engineering 16-1, 31-35, 2014.
2. A. Mohamed A.W.Fathe, A. El-Wahed Hybrid Extreme Learning Machine with Levenberg- Marquardt Algorithm using AHP Method.
3. E.Alba and J. Francisco Chicano, A. El-Wahed Training Neural Networks with GA Hybrid Algorithms.
4. Chen.A hybrid neural network system for prediction and recognition of promoter regions in human genome. 6B (5):401-407 401, 2005.
5. M.I.Ibrahimy, Md. R. Ahsan, O.O.Khalifa Design and Optimization of Levenberg-Marquardt based Neural Network Classifier for EMG Signals to Identify Hand Motions, Measurement Science Review, 13, 142, 2013.
6. A.A.Suratgar, M. B. Tavakoli, A. Hoseinabadi Modified LevenbergMarquardt Method for Neural Networks Training, World Academy of Science, Engineering and Technology, 6, 2005.
7. Koffka Khan, Ashok Sahai ”A Comparison of BA, GA, PSO, BP and LM for Training Feed forward Neural Networks in e-Learning Context”, I.J. Intelligent Systems and Applications,7,23-29, 2012.

8. Amir Abolfazl Suratgar, Mohammad Bagher Tavakoli, and Abbas Hoseinabadi"Modified Levenberg-Marquardt Method for Neural Networks Training", World Academy of Science, Engineering and Technology, 6, 2005.
9. Nazri Mohd Nawi*, Abdullah Khan, M. Z. Rehman"A New Levenberg Marquardt Based Back Propagation Algorithm Trained with Cuckoo Search", ScienceDirect, 18-23, 2013.
10. Ozgur Kisi and Erdal Uncuoglu, "Comparison of three back - propagation algorithms for two case studies", Indian Journal of Engineering Material Sciences, 434-442, 2005.
11. www.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk.../node22.html
12. https://en.wikipedia.org/wiki/Levenberg