IMPACT OF WEIGHT INITIALIZATION ON TRAINING OF SIGMOIDAL FFANN

M.P.S. Bhatia[1], Veenu[2] and Pravin Chandra[3]

[1,2]*Division of Computer Engineering, Netaji Subhas Institute of Technology, India*

[3]*University School of Information Communication and Technology, Guru Gobind Singh Indraprastha University, India*

Abstract

During training one of the most important factor is weight initialization that affects the training speed of the neural network. In this paper we have used random and Nguyen-Widrow weight initialization along with the proposed weight initialization methods for training the FFANN. We have used various types of data sets as input. Five data sets are taken from UCI machine learning repository. We have used PROP Back-Propagation algorithms for training and testing. We have taken different number of inputs and hidden layer nodes with single output node for experimentation. We have found that in almost all the cases the proposed weight initialization method gives better results.

Keywords:

Feed Forward Artificial Neural Network, Back-Propagation Algorithm, Weight Initialization

1. INTRODUCTION

Training of FFANN with Back-Propagation algorithm is an optimization of error with respect to weights. We generally use an algorithm which can reach the local minima in lesser time and which determines the quality and speed of the artificial neural network. Practically the feasible training algorithm may require parameter adjustments for obtaining optimal solution. Also weight initialization is one of most effective approach which is helpful in increasing the training speed. Number of weight initialization methods are proposed like by Drago and Ridella [1], Kim and Ra [2], Sodhi and Chandra [3], Nguyen and Widrow [4], Bhatia and Chandra [5], Stinchcombe and White [6] and many others. These proposed methods are helpful in increasing the convergence speed of the neural network.

Shepherd [7] has discussed the various parameters on which training speed depends. They are training data set, set of non-linear approximation functions, starting initial parameter values of weights, stopping rules or conditions of the algorithm and can train the network by varying all these parameters.

Riedmiller [8] has proposed the RPROP Algorithm for training the artificial neural network. It is resilient propagation algorithm that performs a direct adaptation of the weight step based on local gradient information. The adaptive update value evolves during the learning process based on its local sign on the error function according to the adaptive rule proposed by them.

LeCun [9] has given the efficient BackProp Algorithm for training neural network. He said to choose target values at the point of the maximum second derivative on the sigmoid so as to avoid saturating the output units. The training set should be normalized and weights should be drawn randomly from uniform distribution.

Cherkassky [10] has given the various function estimation methods for generating various input samples. He followed a comparison methodology by choosing flexible method or representation, Learning model parameters, regularization parameters and predictive performance of the final model. He used training set, validation set and test set for training purpose.

Fiesler [11] has performed experiment by varying the shape of the initial weights, variance of the initial weight distribution, order and topology of the network and activation function. He finally after experimentation proposed a suitable weight initialization method for high order perceptron and got better results.

Section 2 describes the BP algorithm and weight initialization method, the various inputs we have taken, various data sets we have taken from UCI machine learning repository, experiment performed, results obtained and finally concluded the work.

2. BACK-PROPAGATION ALGORITHM

Back-Propagation Algorithm is used for training the neural network. It is a very popular algorithm that is used for training. The weights are initialized randomly in the range [-1, 1]. We have used Resillient Back-Propagation algorithm as described below [12] and [13].

Resillient Back-Propagation Algorithm is a training algorithm that updates the weight and bias values according to the resilient back-propagation algorithm RPROP. The purpose of RPROP is to eliminate the harmful effects of the partial derivatives. Only the sign of the derivative can determine the direction of weight update. The magnitude of the derivative has no effect on the weight update. The update value for each weight and bias is increased by a factor of delt inc whenever derivative of performance function w.r.t. weight has the same sign for two successive iterations, decreased by a factor of delt dec whenever derivative w.r.t. weight changes sign from the previous iteration and if the derivative is zero, the update value remains the same.

Whenever the weights are oscillating the weight change is reduced. If the weight continues to change in the same direction for several iterations, the magnitude of the weight change increases. Back-Propagation is used to calculate derivatives of performance w.r.t. weight and bias variables. The training stops when any of the following conditions are met.

- The maximum number of epochs has reached.
- The maximum amount of time is exceeded.
- Performance is minimized to the goal.
- The performance gradient falls below min grad.
- Validation performance has increased more than max fail times since the last time it decreased when using validation.

3. ACTIVATION FUNCTION

Various types of activation functions are described in the books by Haykins [14]. Sigmoidal activation function is used by us for experimentation. It is monotonically increasing function which asymptotes to finite value as infinity is approached. Sigmoid activation function is given by the Eq.(1).

$$f(x) = \frac{1}{(1+e^{-x})} \qquad (1)$$

4. WEIGHT INITIALIZATION

Number of weight initialization methods are given by different researchers [1]-[6] and many more. Most commonly used weight initialization is random initialization between some range [-1, 1] or [-½, ½]. Zero weight initialization is also done by some researchers during training the FFANN. Nguyen-Widrow weight initialization method is also used during training as described by Nguyen-Widrow [4]. We have proposed a new weight initialization method for sigmoidal FFANN in this paper as given below.

```
Algorithm: NEW_wt
    Input:
        NI: number of inputs in input layer.
        NH: number of hidden nodes in the hidden layer.
        NO: number of outputs in the output layer.
    Range for weights: $\lambda_i = \frac{3}{\sqrt{I}}$ for WHI and $\lambda_i = \frac{3}{\sqrt{I}}$ for WHO
    Calculate weights
        WHI: Weight attached with hidden layer node to input layer node ($w_{ij}$).
        WOH: Weight attached with output layer node to hidden layer Node ($\alpha_i$).
    Calculate threshold values
        TH: Threshold of hidden layer ($\boldsymbol{P}_i$).
        TO: Threshold of output layer ($\Upsilon$).
where, $TH = -\sum_{i=1}^{NH}\left(\sum_{j=1}^{NI}\left(w_{ij} * x_j\right)\right)$ and $TO = -0.5\sum_{j=1}^{NI}\alpha_j$
    Set number of epochs
    Call training algorithm RPROP with random and updated inputs.
    Output: minimum MSE, mean of MSE, standard deviation and median of MSE.
```

5. DATA FOR EXPERIMENT

Five different input sets are taken from UCI machine learning repository [16]. They are as described below:

5.1 AIR QUALITY DATA SET

This set consists of 9358 instances of hourly averaged responses from an array of 5 metal oxide chemical sensors embedded in an air quality chemical multisensory device. The data present in this data base is from March, 2004 to February 2005. 12 inputs are taken from this set for inputs in input layer of the ANN. We have removed all the missing values tagged with -200. We got 827 rows which we have normalized between [-1, 1].

5.2 AUTO IMPORTS DATA SET

This set consists of 205 instances. It contains data corresponding to specification of a car. We have considered 12 inputs for our experiment. We have removed all the missing data tagged with '?' and obtained 159 rows of desired data. We have normalized the data set between [-1, 1].

5.3 HOUSING DATA SET

This data set is taken from statlib library which is maintained at mellon university for housing values in suburbs of Boston. We have taken 11 attributes and 506 instances. We have normalized the data in set between [-1, 1].

5.4 CPU PERFORMANCE DATA SET

This data set finds the relative CPU performance based on different parameters. We have considered 7 attributes and 209 instances. This data set is normalized between [-1, 1].

5.5 CONCRETE COMPRESSIVE STRENGTH DATA SET

This set consists of data corresponding to actual concrete compressive strength. We have taken 8 attributes. Total 1030 instances are taken and normalized between [-1, 1]. MATLAB R2013a is used for coding and MSE is calculated using all these data sets as described above. We have used one input layer, one hidden layer and one output layer. The number of inputs in the input layer and hidden layer varies. We have considered single node in the output layer. The network size considered for our experiment is shown in the Table.1 given below.

Table.1. Architecture Used

No. of inputs	No. of outputs	No. of hidden neurons	Data set used
12	1	26	Air Quality data
12	1	21	Auto Import data
11	1	25	Housing data
7	1	16	CPU performance data
8	1	27	Concrete Compressive strength data

6. RESULTS

We have implemented the algorithm in MATLAB R2013a 8.1. We have taken five data sets from UCI repository. They are:

- Air Quality Data set having size of each input data item of 827.
- Auto Import data set having size of each input data item of 159.
- Housing data set having size of each input data item of 506.
- CPU performance data set having size of each input data item of 209.
- Concrete compressive strength data set having size of each input data item of 1030.

These sets of input are divided further into two sets. 70% data is used for training and 30% data is used for testing. Total 1 to 30 networks are used for training and testing. Weights are initialized using three methods. They are:

- Random weight initialization: Rand_wt.
- Nguyen-Widrow weight initialization: NW_wt.
- Proposed weight initialization method NEW_wt.

The results are shown in the Table.2. The various notations used are:

- Rand_wt – Random weight initialization.
- NW_wt – Nguyen-Widrow weight initialization
- NEW_wt – Proposed weight initialization
- Min_MSE – Minimum MSE
- Mn_MSE – Mean MSE
- Md_MSE – Median MSE
- Std_DVE – Standard deviation

7. CONCLUSION

We have observed that the training of FFANN depends on number of parameters. Its output depends on the initial weights we define. In this paper we have initialized weights using random, Nguyen_Widrow, NEW_wt methods. NEW_wt is the proposed weight initialization method. Implementation is done using MATLAB R2013a. Five different data sets are taken. These five data sets are taken from UCI machine learning repository. We have considered one input layer, one hidden layer and one output layer. Number of inputs vary in the input layer and the hidden layer. We have considered single node in the output layer. The data sets are divided into 70% training set and 30% testing set. The sets are normalized between the range [-1, 1]. RPROP algorithm is used for training and testing. We have compared all the outputs obtained after execution. The results obtained shows that when we have taken air quality data set then we are obtaining minimum error. Hence we have best results in this case. Also we have found better results by using our proposed method of weight initialization when compared with other two methods of weight initialization. In this paper we have used only one training algorithm. We are going to compare the results with other training algorithms in future.

Table.2. Comparative table for the outputs obtained using RPROP Algorithm.

Data set used	**Statistics**	**Training**			**Testing**		
		Rand_wt Init.	**NW_wt Init.**	**NEW_wt Init.**	**Rand_wt Init.**	**NW_wt Init.**	**NEW_wt Init.**
Air Quality data	Min_MSE	0.00034418	0.0002901	0.0001324	0.0005156	0.0003812	0.00021755
	Mn_MSE	0.00077318	0.0004491	0.0003009	0.0012129	0.0007469	0.00046658
	Md_MSE	0.0007859	0.0004167	0.0003039	0.001170	0.0006495	0.00044853
	Std_ DVE	0.0001895	0.0001215	0.0000805	0.0003751	0.0003140	0.00016138
Auto Import data	Min_MSE	0.005792	0.007461	0.004152	0.017423	0.022331	0.010591
	Mn_MSE	0.014162	0.015087	0.0115258	0.044273	0.047564	0.035867
	Md_MSE	0.013651	0.013378	0.009583	0.042279	0.041992	0.029449
	Std_ DVE	0.0058078	0.0065549	0.0058039	0.019085	0.021418	0.019142
Housing data	Min_MSE	0.0101357	0.010316	0.009079	0.022086	0.022484	0.017606
	Mn_MSE	0.013088	0.0125584	0.011865	0.028017	0.026409	0.025822
	Md_MSE	0.012552	0.012388	0.011356	0.025637	0.026500	0.023399
	Std_ DVE	0.002019	0.001375	0.002047	0.006589	0.004372	0.007805
CPU Performance data	Min_MSE	0.0001669	0.0001841	0.000109	0.0004256	0.0004938	0.0002483
	Mn_MSE	0.001519	0.004291	0.001791	0.004794	0.014065	0.0057875
	Md_MSE	0.0006533	0.002290	0.001065	0.002004	0.007434	0.0033359
	Std_ DVE	0.0021072	0.004756	0.002492	0.006963	0.015795	0.0082695
Concrete Compressive strength data	Min_MSE	0.0172478	0.014285	0.014798	0.024045	0.019148	0.020504
	Mn_MSE	0.0205826	0.017486	0.018422	0.030081	0.026476	0.027253
	Md_MSE	0.02001	0.017539	0.018168	0.0301417	0.025726	0.026668
	Std_ DVE	0.0021297	0.001848	0.001587	0.0042162	0.0047455	0.003879

REFERENCES

[1] G.P. Drago and S. Ridella, "Statistically Controlled Activation Weight Initialization (SCAWI)", *IEEE Transactions on Neural Networks,* Vol. 3, No. 4, pp. 627-631, 1992.

[2] Y.K. Kim and J.B. Ra, "Weight Value Initialization for Improving Training SPEED in the Back Propagation Networks", *Proceedings of IEEE International Joint Conference on Neural Networks,* pp. 2396-2401, 1991.

[3] S.S. Sodi and P. Chandra, "Interval based Weight Initialization Method for Sigmoidal Feed Forward Artificial Neural Networks", *Proceedings of 2nd AASRI Conference on Computational Intelligence and Bioinformatics*, pp. 19-25, 2014.

[4] D. Nyugen and B. Widrow, "Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights", *Proceedings of International Joint Conference on Neural Networks,* pp. 23-29, 1990.

[5] M.P.S. Veenu Bhatia and P. Chandra, "Comparison of Sigmoidal FFANN Training Algorithms for Function Approximation Problems", *Proceedings of International Conference on Computing for Sustainable Global Development*, pp. 325-329, 2015.

[6] M. Stinchcombe and H. White, "Approximating and Learning Unknown Mappings using Multilayer Feed Forward Networks with Bounded Weights", *Proceedings of International Joint Conference on Neural Networks,* Vol. 3, pp. 7-16, 1990.

[7] V. Cherkassky and R. Shepherd, "Regularization Effect of Weight Initialization in Back Propagation Networks", *Proceedings of IEEE International Joint conference on Neural Networks*, pp. 2258-2261, 1998.

[8] M. Reidmiller and H. Braun, "A Direct Adaptive Method for faster Back Propagation Learning: The RPROP Algorithm", *Proceedings of IEEE International Conference on Neural Networks*, pp. 586-591, 1993.

[9] Y. LeCun, L. Bottou, G. Orr and K. Muller, "*Efficient BackProp*", Springer, 1998.

[10] V. Cherkassky, Do Gehring and F. Mulier, "Comparison of Adaptive Methods for Function Estimation from Samples", *IEEE Transactions on Neural Networks,* Vol. 7, No. 4, pp. 969-984, 1996.

[11] G. Thimm and E Fiesler, "Neural Network Initialization", *Proceedings of International Workshop on Artificial Neural Networks*, pp. 535-542, 2005.

[12] MATLAB version R2013a 8.1, Available at: https://www.mathworks.com/matlabcentral/answers/112649-matlab-student-r2013a-compatibility-with-windows-8-1.

[13] Neural Network Toolbox, Available at: http://www.mathworks.in/help/nnet/ref.

[14] Simon Haykin, "*Neural Networks-A Comprehensive Foundation*", 2nd Edition, Prentice Hall, 1999.

[15] Simon Haykin, "*Neural Networks and Learning Machines*", 3rd Edition, PHI Learning Private Limited, 2011.

[16] UCI Machine Learning Repository, Available at: archive.ics.uci.edu/ml/.