2011 International Conference on Information and Electronics Engineering
IPCSIT vol.6 (2011) © (2011) IACSIT Press, Singapore

Applying Multiple Neural Networks on Large Scale Data

Kritsanatt Boonkiatpong [1] and Sukree Sinthupinyo [2]

[1] Department of Computer Engineering Chulalongkorn University, Bangkok, Thailand
E-mail: g51akl@cp.eng.chula.ac.th

[2] Department of Computer Engineering Chulalongkorn University, Bangkok, Thailand
E-mail: sukree.s@chula.ac.th

Abstract. Analysis on large data sets is highly important in data mining. Large amount of data normally requires a specific learning method. Especially some standard methods, for example the Artificial Neural Network, need very long learning time. This paper presents a new approach which can work efficiently with the neural networks on large data sets. Data is divided into separated segments, and learned by a same network structure. Then all weights from the set of networks are integrated. The results from the experiments show that our method can preserve the accuracy while the training time is dramatically reduced.

Keywords: Neural Network, Large Scale Dataset, Incremental Learning

1. Introduction

Studies in the neural network have been divided into several aspects whether it is a study in structure modelling, network design, and performance improvement to quickly learn and to achieve more accurate results [1]. There are many neural network related techniques including applications in various fields, for example, data mining, image recognition, weather forecasting, traffic, stock market, etc. Research in neural network is also aimed to improve in different ways including faster network processing, more efficiency, or fewer errors. These are still a focus of research attention.

This research will focus on the analysis of large data by applying multiple neural networks to learn several sub dataset. Shibata and Ikeda showed that the number of neurons and the number of hidden layers in the network can affect performance [2], because a small number of layers can process faster than a big one. Their work focuses on the structure level of the neural network. Generally, number of hidden layer can increase the accuracy of learning. But it will affect the learning time much more than a small layer. In addition, a large data set is difficult to learn at one time. It requires both resources and time. Thus, this paper presents an idea that we can separate the large data sets to be multiple subsets each of which is trained using same structure neural networks. Then we improve the overall accuracy using technique that replaces the appropriate weight from the smallest error to each node of the neuron. Then, the weights of the best small data sets to are used to create a new network. The purposed technique help the neural network learn larger data sets. Using our purposed technique, the accuracy is comparable to the network trained by the whole data set while the training time is dramatically decreased.

2. Backpropagation Neural Network

In this research, a famous learning technique, i.e. back-propagation, will be used to train a neural network. Backpropagation is a supervised learning technique, in which neurons connect each other with weighted connection. A signal transmitted to the next neuron is weighted by a link connecting from one node to another node as shown in (Fig. 1).

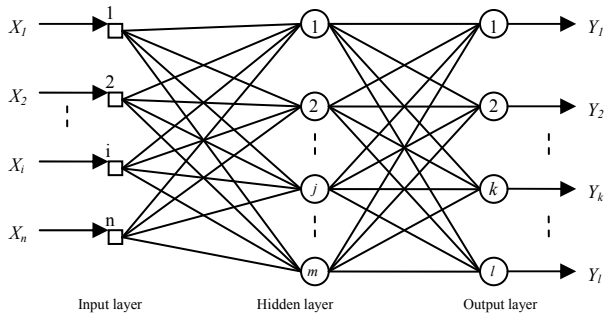

Fig. 1: Backpropagation Neural Network Structure

Weight and threshold will be determined in a transfer functions. It will be transmitted by the neuron in the hidden layer. Its output is given by (1).

$$Y_j(P) = sigmoid\left(\sum_{i=1}^{n} X_i(P) \times W_{ij}(P) - \theta_j\right) \quad (1)$$

P is a learning example in a dataset, ***n*** is number of input of ***j*** neurons in a hidden layer, X_i is input ***i*** that transmit to neuron and Y_j is output, ***W*** is weight of each neuron and θ_j is a threshold. The sigmoid function, is shown in (2).

$$sigmoid(x) = \frac{1}{1+e^{-x}} \quad (2)$$

The results from the calculation will be fed to output layer using the function below.

$$Y_k(P) = sigmoid\left(\sum_{j=1}^{m} X_{jk}(P) \times W_{jk}(P) - \theta_k\right) \quad (3)$$

Where ***P*** is a learning example in dataset, ***m*** denotes the number of neuron in output layers, ***k*** shows the output layer, X_{jk} is input from hidden layer, and ***Y*** is output. ***W*** is weight of each neuron, and θ_k is a threshold of this output node.

3. Methodology

3.1. Training Separation

Our experiment is aimed at improving the original training algorithm to be able to train a large dataset by separated equally sub datasets. The idea behind the proposed method is that the smaller sub data set will consume less training time than a big one. However the error from each sub data set is unstable and all weights are separated. Hence, we will collect the weight from each node in the lowest error network and use these weights to replace the trained weight of other networks with the same structure.

Assume that we divide the original training set (N_0) into *n* sub data sets, namely N_1 to N_n. Each equally separated set will contain ***X/n*** instances and each sub set will be trained by the same BP network structure with a single hidden layer. The concept of our training method is showed if Fig.2.

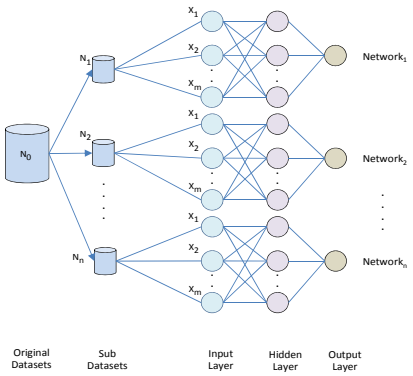

Fig. 2: Training method which divides the original datasets into ***n*** sub data sets each of which will be used as training sets for ***n*** BP network**s.**

When all networks are completely trained, we will evaluate results of each network. The weight from the best trained network (N_{best}) will be used as a starting weight set in the weight integration process.

N_{best} will be applied to all other datasets to clarify that N_{best} has better result or the network trained from of the set itself is better than N_{best}. During this process, the weight replacement which will be described in section 3.1 is employed.

After all network comparisons and weight integrations and completed, we will have a neural network with the same structure as the original, but the weights have been chosen to best suited the all sub data sets.

3.2. Network Integration Method

We change the weights in N_{best} using two methods, i.e. Weight Combination and Node Creation, The first method is used when two hidden nodes from different networks are closed to each other. The new weights of one node in N_{best} will be set as the average weights from both hidden nodes as shown in Equation (1).

$$W_{N_{best}} \leftarrow (W_{N_{best}} + W_{N_i})/2 \quad (1)$$

where, $W_{N_{best}}$ and W_{N_i} denote the weight vector of two hidden nodes which are closed to each other.

In the latter case, when a node from N_i cannot be combined to N_{best}, that node is directly inserted into N_{best}.

4. Experimental Results

4.1. Data Preparation

In this experiment, we used two data sets from UCI repository [9]. Both are Iris and Letter Recognition. The Letter Recognition contains 20,000 records which take very long learning time using the ordinary training method. We used a machine learning tool, WEKA [10], and MBP [11] in all of our experiments.

4.2. Network Convergence

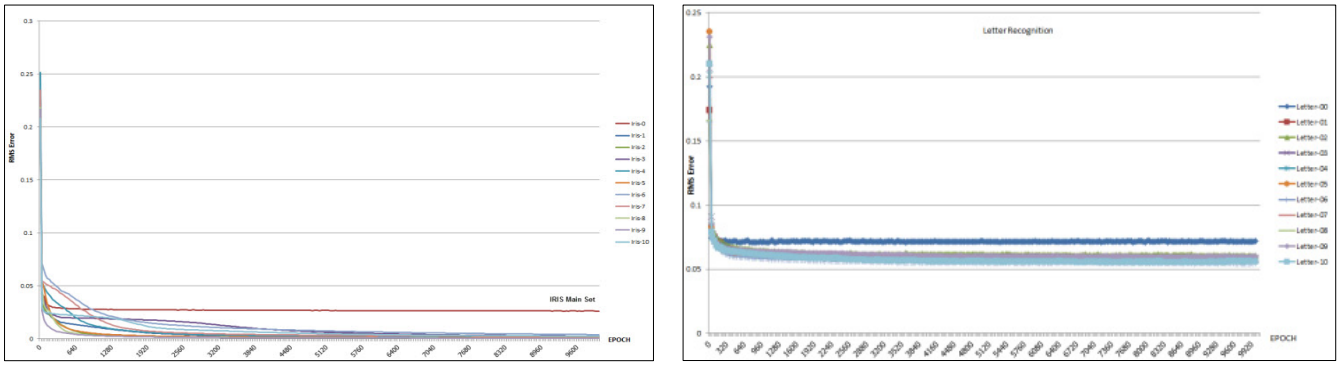

Fig. 3: Error convergence of Iris (left) and Letter Recognition (right) dataset

Fig. 3 shows error of dataset, **Iris-0** indicates the main dataset, **and Iris-1 to Iris-10** denotes ten sub datasets. The right graph shows error on Letter Recognition dataset. **Letter-01** to **Letter-10** are error from sub dataset 1 to 10 respectively. **Letter-00** is the original training set. We can see from both graphs that the error of the sub datasets is much lower than the error of the original training set. The trained weights converge to the point that gives the lower error rate on the training set.

4.3. Experimental Results

In this section, the accuracy of sub datasets from both datasets is shown in Table 1. N_9 and N_6 were the best networks from Iris and Letter Recognition, respectively. After we obtained the best network (N_{best}), we could integrate weights from all networks and the results and the results are shown in Table 2.

Table 1: RMS Error of each network and main network on its own dataset

	RMS Error	
Data Set	**Iris**	**Letter Recognition**
N_{all}	0.0110874657	0.0720147465
N_1	0.0018587002	0.0571487767
N_2	0.0009121711	0.060338648
N_3	0.004561998	0.0585201252
N_4	0.0011229377	0.0576738189
N_5	0.0038304094	0.0569459839
N_6	0.0038304094	**0.0547496419**
N_7	0.0049999925	0.0551124568
N_8	0.0009161137	0.0581173707
N_9	**0.0007008063**	0.0596956224
N_{10}	0.0032190801	0.0556769854

Table 2: RMS Error of N_{best} before and after integration

Dataset	Weight Set	RMS Error
Iris	N_{best}	0.0110874657
	N_{best} (after integration)	0.0103989667
Letter Recognition	N_{best}	0.0720147465
	N_{best} (after integration)	0.0672054431

5. Conclusion

This paper has proposed a new method which can apply Backpropagation Neural Network to large datasets. We can see from our experiment that weights of small dataset converged faster than the original dataset. However, the weights trained from sub dataset did not achieve better result when they were tested on the original dataset. Hence, our weight integration approach was introduced. After the rule integration, we found that the accuracy of the weight obtained from the proposed method is better than the weight set which is the best among the sub datasets.

6. References

[1] Y. Zhao, J. Gao, and X. Yang, A Survey of neural network ensembles, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2005.

[2] K. Shibata and Y. Ikeda, Effect of number of hidden neurons on learning in large-scale layered neural networks, *ICROS-SICE International Joint Conference*, 2009, p5008-5013.

[3] K. Kaikhah and S. Doddameti, Discovering Trends in Large Datasets Using Neural Networks, *Applied Intelligence, Springer Science + Business Media, Inc.*, Netherlands, vol. 24, 2006, pp. 51–60.

[4] B. Liang and J. Austin, A neural network for mining large volumes of time series data, *IEEE Transactions on Neural Networks*, 2005, pp.688-693.

[5] L. Fu, H. Hsu, and J. Principe, Incremental Backpropagation Learning Networks, *IEEE TRANSACTIONS ON NEURAL NETWORKS, vol. 7, no. 3*, 1996, pp. 757-761.

[6] D. Xia, F. Wu, X. Zhang, and Y. Zhuang, Local and global approaches of affinity propagation clustering for large scale data, *Journal of Zhejiang University SCIENCE*, 2008, p1373-1381.

[7] J. Heaton, Introduction to Neural Networks for C#, Second Edition, *Heaton Research, Inc., Chesterfield, St. Louis, United States, Second Edition*, 2008, pp. 137-164.

[8] N. Lopes and B. Ribeiro, Hybrid Learning in a Multi Neural Network Architecture, *IEEE Transactions on Neural Network, CISUC- Centro de Informatica e Sistemas, Department of Informatics Engineering, University of Coimbra, Portugal*, 2001, pp. 2788-2793.

[9] UCI Data Sets, The UCI Machine Learning Repository, *Center for Machine Learning and Intelligent Systems, University of California, Irvine, United States*, 2007, http://archive.ics.uci.edu/ml/datasets.html

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations, Volume 11, Issue 1,* 2009.

[11] Multiple Back-Propagation, Back-Propagation Simulation Tool, *Noel de Jesus Mendonça Lopes Instituto Politécnico da Guarda, Portugal*, 2009, http://dit.ipg.pt/MBP