# MapReduce-based Backpropagation Neural Network over Large Scale Mobile Data*

Zhiqiang Liu, Hongyan Li†, Gaoshan Miao
Key Laboratory of Machine Perception (Peking University), Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
Beijing 100871, P. R. China
{liuzq, lihy, miaogs}@cis.pku.edu.cn

***Abstract*—Large scale mobile data are generated continuously by multiple mobile devices in daily communications. Classification on such data possesses high significance for analyzing the behaviors of mobile customers. However, the natural properties of mobile data presents three non-trivial challenges: large data scale leads it difficult to keep both efficiency and accuracy; similar data increases the system load; and noise in the data set is also an important influence factor of the processing result. To resolve the above problems, this paper integrates conventional backpropagation neural network in the cloud computing environment. A MapReduce-based method called MBNN *(i.e. MapReduce-based Backpropagation Neural Network)* is proposed to process classifications on large-scale mobile data. It utilizes a diversity-based algorithm to decrease the computational loads. Moreover, the Adaboosting mechanism is introduced to further ameliorate the performance of classifications. Extensive experiments on gigabyte of realistic mobile data are performed on a cloud computing platform. And the results show that MBNN is characterized by superior efficiency, good scalability and anti-noise.**

***Keywords-MapReduce; large-scale data; Backpropagation neural network; Adaboosting***

## I. INTRODUCTION

Classification of the mobile data generated by the mobile communication customers in the mobile community is very useful for mobile operators because the results can be used to analyze the behaviors of the customers. However, large scale mobile data are widely produced in mobile community, especially with the increase of customers and applications recently. Thus, to analyze large-scale mobile data with scalable conventional data mining methods has become an important research issue for mobile operators as well as database community.

Conventional data mining algorithms for classification (e.g. backpropagation neural network) are not suitable for the cloud computing platform of China Mobile Research Institute (CMRI) based on MapReduce framework [1]. Moreover, when the data scale is large (usually in the size of gigabytes), such an algorithm might be very slow and sometimes cannot even run out a result at all. So this paper studies a conventional data mining algorithm widely used, backpropagation neural network, and proposes a MapReduce-based framework on cloud computing platform in order to improve the efficiency and scalability over large scale mobile data. This method is called MBNN (i.e. **M**apReduce-based **B**ackpropagation **N**eural **N**etwork).

Google's MapRedcue framework is well known as a parallel programming model for cloud computing. It is built on the top of a distributed file system and supports the parallelization of data processing on large clusters. However the research of how to design a neural network on MapReduce framework is rarely touched nowadays especially over large scale mobile data.

Although the MBNN can process large scale data, the efficiency of the method and precision of the network model may not be assured since the data scale is so large. During the study, several natural properties of data are observed, which can affect the training process of a neural network. Accordingly, the diversity-based and Adaboosting methods are introduced in this paper to improve the efficiency of the method and the precision of classification in terms of these natural properties.

Based on our observation, the data have the following natural properties: large scale, similarity and noisiness etc. Firstly, the mobile data is large-scale because the China Mobile has large number of customers the cardinality of which is over one hundred million. The information related to these customers is also in large scale and usually in size of terabytes. And these customers induce data generation everyday as long as they are active. Secondly, lots of the data are similar and noisy with each other. The reason might be that many customers have similar behaviors. When these similar data participate in the classification procedure of backpropagation neural network, they contribute little to the precision of classification and may disturb the training process of the algorithms.

Since the training sets of the neural network are enormous and noisy, the precision of the model is not high enough. Eventually, the precision of the model over large scale mobile data sets is usually about 50% which meets the definition of weak learning [2]. And it poses a challenging study in large scale mobile data Hence, in this paper, we introduce Adaboosting [3] method to improve the precision of the neural network.

† Hongyan Li is the corresponding author
*This work is supported by Natural Science Foundation of China (NSFC) under grant numbers 60973002 and 60673113

The following is the organization of this paper. Section 2 discusses related work. Section 3 presents the details of our proposed MBNN method. Section 4 describes the experiments which prove the efficiency, anti-noise and scalability of our proposed method. And Section 5 makes a conclusion of this paper and discusses the future work.

## II. RELEATED WORK

### A. Backpropagation and mobile data processing

Backpropagation [4] (BP) is one of the most widely used algorithms for supervised learning with multi-layered feed-forward neural networks [5]. The weights update in BP algorithm has two modes: the online mode and the batch mode. The batch update mode is more suitable for MapReduce framework of the cloud computing platform because all the items in the training set can be used concurrently. Backpropagation and its improved methods are applied in mobile data processing. Such applications can be found in [6][7][8]. But their mobile data sets for experiments are not in large scale. The size of their data is only megabyte-class. In contrast, our method can process mobile data in gigabyte-class.

### B. Parallel method for neural network

There are some parallel methods of neural network for the cluster computing. These methods achieve the parallelization of the algorithm by mapping the network nodes onto the nodes of clusters. The methods proposed by [9][10] and [11] adopt this idea. But in the cloud computing environment based on MapReduce, the programmer can't control one special cluster node to do certain jobs. The distribution of the network nodes can't be controlled by the programmer, so these methods are not implementable in the cloud computing environment. Moreover, the amount of computation on each network node is small. Mapping the nodes onto different computers increases the I/O costs. In most cases it is not cost-efficient for the reason that the computation is I/O-oriented in this environment. It refers that the I/O cost is the main cost in the distributed environment. Thus，those parallel methods cannot be applied effectively in a cloud computing environment.

Chu et al. [12] proposed a method which used Backpropagation algorithm to implement a three layer neural network with two output neurons classifying the data into two categories based on MapReduce and did some experiments on multicore. But he didn't apply the method to process large-scale mobile data on large cloud clusters.

### C. Adaboosting method

The Adaboosting method is a general optimization method for classification. It can improve the performance of a certain classifier by assembling several classifiers to form a classifier system. Since the data is large-scale and noisy, the precision of the classifier is always not high and is conform to weak learning problem. Thus performance of the classifier can be improved by applying the Adaboosting method.

## III. MBNN: MAPREDUCE-BASED BACKPROPAGATION NEURAL NETWORK

A MapReduce-based backpropagation neural network (MBNN) is proposed by this paper to make conventional BP neural network work effectively in the cloud computing platform of CMRI. Our method has three kernels: implementation of MBNN, diversity-based data sampling method, and Adaboosting neural network method. In respect that the platform is MapReduce-based, the method should be thoroughly implemented based on the MapReduce framework. Since the data of CMRI have lots of similar items, the diversity-based sampling method is introduced. The Adaboosting method is adopted to improve the precision of classification for the reason that the precision of the classification over the large scale mobile data is not high enough.

The following part describes the detailed introduction of the three kernels of our method.

### A. Implementation of MBNN

Conventional backpropagation algorithm is used as the basic algorithm for this implementation with some thinking referencing [13].

MapReduce is a programming model and an associated implementation for large scale mobile data process. Programmers only need to specify a MapReduce job which is composed of the mapper and reducer functions. A mapper function receives a key/value pair and generates a set of intermediate key/value pairs. And a reducer function merges all intermediate values associated with the same intermediate key. Once the mapper and reducer functions are specified, a MapReduce job is defined.

The following is the basic procedure of this method:

A. Receive many items of the training set.

B. Execute several mapper tasks. Each mapper receives one training item and then computes all update-values for the weights in the network using this training item.

C. Execute several reducer tasks. Each reducer gathers update-values for one weight and calculates an average value of these values. Then it outputs the average value as the update-value for the weight.

D. Do a batch update on all the weights in the network.

E. Execute step B, C and D until the expected precision is achieved.

Details of the mapper and reducer functions can be found in procedure 1 and procedure 2:

**Procedure 1 Backpropagation Mapper**

| | |
|---|---|
| 1 | Input key/value pair < *key*, *value* = inputItem >. |
| 2 | Compute the update-value for all the weights using *value* as the input of the network |
| 3 | For every weight $w$, get its weight $\Delta w$ |

| | |
|---|---|
| 4 | Emit intermediate key/value pair <*w*, *Δw*> |

**Procedure 2 Backpropagation Reducer**

| | |
|---|---|
| 1 | Input key/value pair < *key* = *w*, *value* = *Δw* >. |
| 2 | *sum* ←0, *count* ← 0 |
| 3 | *sum* ← *sum* + *value* |
| 4 | *count* ← *count* + 1 |
| 5 | **If** more pairs <*key*, *value*> are collected, go to step 1<br>**Else** output key/value pair <*key*, *sum/count*> |

The main function is designed as following procedure 3:

**Procedure 3 Backpropagation Main Function**

| | |
|---|---|
| 1 | Run the job whose mapper and reducer functions are the backpropagation mapper and reducer in procedure 1 and 2. |
| 2 | Compute the output value of each reducer and do a batch update on weights of the network |
| 3 | **If** the precision of the network on the training set is not better than the expected precision, go to step 1<br>**Else** procedure end. |

Below is analysis of time complexity for this method, and we also analyze the scalability of our proposed method.

One item of the training set corresponds to one mapper task. As a result the number of mapper tasks is the same as the number of items in the training set. The number of the reducer tasks is the same as the number of the weights.

The data items can be seen as vectors. Each dimension of the vector is one attribute of the item. Suppose that the number of attributes participated in the classification is *v*, and the number of the neural network inputs is also *v*. The number of result classes of the classification is *o*, and the number of the neural network outputs is *o*. The number of the middle layers is *L*. The number of nodes of the *i*th middle layer is $K_i$ ($i = 1 \ldots L$). Suppose the arcs of the neural network is *a*, then *a* can be calculated by formula (1).

$$a = vK_1 + \sum_{i=1}^{L-1} K_i K_{i+1} + oK_L \qquad (1)$$

Now suppose the number of the training items is *n* and the number of mapper and reducer nodes is *m* and *r*. Because each map task must compute the update-values for all weights, the complexity of map task is $O(a)$. There are *n* map tasks to be process by *m* mapper nodes parallel. So the whole complexity is $O(an/m)$. And each reduce procedure must calculate the average value of the *n* update-values, so the complexity of reduce task is $O(n)$. There are *a* reduce tasks to be process by *r* reducer nodes parallel. So the whole complexity is $O(an/r)$. Then the time complexity of one loop of backpropagation is $O(an/m)+O(an/r)$. If the loop number is N, the complexity of the whole backpropagation training method is $O(anN/m)+O(anN/r)$.

Scalability is an important indicator to evaluate methods over large scale mobile data in distributed environment. If a method has an excellent scalability capability, the efficiency of the method should be improved *x* times at least if the number of the nodes in the cluster is increased by *x* times.

When the number of nodes in the cluster increases *x* times, the number of mapper and reducer nodes also increases *x* times. Then the complexity becomes $an/xm + an/xr$. So the complexity of the algorithm becomes $1/x$ of the original complexity. Thus this method performs well in terms of scalability.

### *B. Diversity-based data sampling method*

There are lots of similar items in the mobile data. In order to improve the efficiency of the training, a sampling method is introduced to remove these similar items. The training set after sampling can be more typical and can represent the feature of the data set better. It can decrease the training loop number so as to save the training time.

The following method can be used to eliminate similar items. First, run a MapReduce job called *RangCount* to count the ranges of each dimension of the items in the training set. And then divide the range of each dimension into *p* portions averagely. Second, for each item in the training set, get the feature vector for it. Last, run a job called *DiversitySampling* to aggregate the items with the same feature vector into one group. One group is sampled out only one item randomly. Those sampled items compose the new training set.

Details of mapper and reducer functions of the *RangeCount* job are introduced in the following procedure 4 and 5:

**Procedure 4 RangeCount Mapper**

| | |
|---|---|
| 1 | Input key/value pair < *key*, *value* = inputItem >. |
| 2 | For each dimension of the item, emit an intermediate key/value pair <*dim*, *value*>. *dim* is the dimension number and *value* is the value of this dimension. |

**Procedure 5 RangeCount Reducer**

| | |
|---|---|
| 1 | Input key/value pair < *key* = dim, *value* >. |
| 2 | Initialize *Min* ← -MAX, *Max*← MAX |
| 3 | **If**(*value* < *Min*) *Min* ← *value*<br>**Else If**(*value* > *Max*) *Max* ← *value* |
| 4 | **If** more pairs <*key*, *value*> are collected go to step 1<br>**Else** Output key/value pair <-*dim*, *Min*> and <*dim*, *Max*> |

After the job with this Mapper and Reducer function, the minimum value of a dimension is the value of the pair with the key *–dim* and the maximum value is the value of the pair with the key *dim*.

The *i*th dimension of feature vector of an item can be computed by the following formula (2):

$$F_i = \lfloor p(v - min_i) / (max_i - min_i) \rfloor \qquad (2)$$

*p* is the portions divided on the dimension and *v* is the value of the *i*th dimension of the item. The maximum and minimum value of the dimension is $max_i$ and $min_i$. *d* is the number of dimension. Then the feature vector for this item is $< F_1, F_2, ..., F_d >$.

This sampling procedure can also be implemented by MapReduce job. This job is called *DiversitySampling*. The mapper and reducer functions of this job are described in the following:

**Procedure 6 DiversitySampling Mapper**

1 Input key/value pair < *key*, *value* = inputItem >.

2 Compute the feature vector $F =< F_1, F_2, ..., F_i >$.

3 Emit intermediate key/value pair <*F*, *value*>

**Procedure 7 DiversitySampling Reducer**

1 Input key/value pair < *key* = *F*, *value* >.

2 Collect all the pairs < *key*, *value* > and sample out one pair from those pairs randomly. The value of sampled pair is *v*.

3 Output key/value pair <*k*, *v*>, *k* can be an arbitrary value since it is no use.

### C. Adaboosting method for neural network

The neural network model after training can be regarded as a classifier. The precision of the classifier is always not high enough due to the noisy and the large-scale of the mobile data. Sometimes it is only about 50%. Such learning is called weak learning. In order to improve the performance of the classifier, the boosting method is introduced.

The most widely used boosting method is the Adaboosting method. A weak classifier is a classifier whose precision is just better than 50%. The basic concepts of the Adabooting method can be summarized as following:

A. Get one weak classifier using part of the training set.

B Get more weak classifiers using different parts of the training set sampled out by the features of the former classifiers.

C. Assemble those classifiers to form a classifier system.

Generally, this method builds up a set of classifiers and each classifier's training set is built up by the most informative items for the former classifiers. The final classifier result is determined by the results of those classifiers together. The Adaboosting method can be used recurrently. So the designer can add new weak classifiers to the classifier system until the system achieve the required precision.

The detailed discussion and procedure of Adaboosting method for neural network can be found in [3] and [14].

## IV. EXPERIMENTS

In this section, extensive experiments are conducted and the experimental results prove the efficiency, anti-noise, and scalability of the method over large scale realistic mobile data on the cloud computing platform of CMRI.

### A. Experiment environment

The CMRI cloud computing platform is build up by 16, 32, 64 and 128 nodes etc. Data in size of gigabytes and terabytes are stored distributed in the platform. Each node is equipped with the Intel(R) CPU 2.2GHZ, 8 G memory and 4*250GB 7200r/s hard disks. The nodes are connected with each other by gigabit Ethernet. The operating system of the nodes is Red Hat Linux. The open source framework Hadoop [15] is adopted for the distributed organization of those nodes.

### B. Efficiency and scalability test

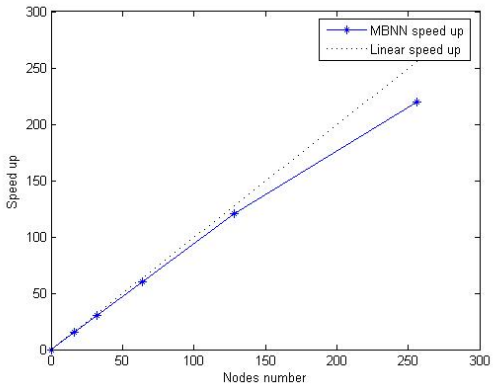


Figure 1. Speed up of MBNN on different number of nodes.

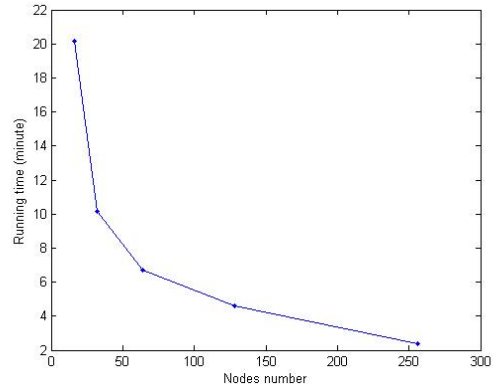


Figure 2. Running times on different number of nodes.

The efficiency and scalability of the MBNN method is tested on cloud clusters with 16, 32, 64, 128 and 256 nodes. The training set size is 13.46GB. The time in figure 1 is the time of 4 loops of batch training. Figure 1 shows that the speed up of MBNN for conventional serial neural network is only a

bit worse than linear speed up. It proves that MBNN have an excellent speed up performance. Figure 2 shows that the time has an inverse ratio relationship with the nodes number approximately, so that the method has an excellent scalability. That accords with the scalability analysis in section 3

### C. Diversity-based sampling test

This test is operated on a cloud cluster of 64 nodes. TABLE Ⅰ gives the results of the diversity-based sampling method applied to the training sets of different size. The original data scale is the scale of training set. The data scale after sampling and the sampling rate is shown in this table. When the data scale is decreased, the time of one loop of training is also decreased. Thus the training time is saved.

Suppose the original data scale is *OS*, and the data scale after sampling is *SS*. Then the sampling rate *SR* can be calculated by the follow formula (3):

$$SR = SS / OS \tag{3}$$

TABLE I. DIVERSITY-BASED SAMPLING TEST

| Original data scale | Data scale after sampling | Sampling rate |
|---|---|---|
| 13.46GB | 11.20GB | 0.732 |
| 26.92GB | 22.24GB | 0.746 |
| 53.84GB | 51.33GB | 0.836 |
| 107.67GB | 96.04GB | 0.793 |

TABLE II. PRECISION AFTER SAMPLING TEST

| Original data scale | Precision on original data set | Precision on sampled data set |
|---|---|---|
| 13.46GB | 0.632 | 0.748 |
| 26.92GB | 0.691 | 0.780 |
| 53.84GB | 0.713 | 0.791 |
| 107.67GB | 0.763 | 0.825 |

TABLE Ⅱ shows that the precision on the sampled data set is higher than the original mobile data set because the similar data are eliminated. The precision is got by applying the neural network model on test set which is 500.35GB.

The above results prove that this method is anti-noisy since it can save the time and improve the precision of the training.

### D. Adaboosting test

In this test, the node number of the cluster is also 64. TABLE Ⅲ is the Adaboosting test results. The precision of the classification increases as the number of the weak classifiers increases. So the precision of classifier can be improved by the Adaboosting method.

TABLE III. ADABOOSTING TEST

| Weak classifier number | 1 | 2 | 3 |
|---|---|---|---|
| Precision | 0.553 | 0.692 | 0.870 |

## V. CONCLUSIONS

In this paper, a MapReduced-based backpropagation neural network (MBNN) over large scale mobile data was proposed in the context of cloud computing platform. To improve efficiency of the training process as well as the precision, the diversity-based and Adaboosting methods are introduced. Extensive experiments were conducted based on gigabyte-class mobile data sets, and the experimental results proved the efficiency, scalability and anti-noise of our proposed method.

In the near future, we plan to design and implement more data mining algorithms on the CMRI cloud computing platform in order to make a further support for the mobile operators.

## ACKNOWLEDGMENT

The China Mobile Research Institute (CMRI) is appreciated for providing cloud computing platform as well as data sets for the experiments.

## REFERENCES

[1] J. Dean and S. Ghemawat, “MapReduce: Simplified data processing on large clusters”, ACM OSDI, 2004.

[2] Valiant LG, “A theory of learnable”, Communications of the ACM, 1984.

[3] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, “Boosting the margin: A new explanation for the effectiveness of voting methods”, In Machine Learning: Proceedings of the Fourteenth International Conference, 1997.

[4] S. Haykin, “Neural Networks: A Comprehensive Foundation”, 1994

[5] R. Hecht-Nielsen, “Theory of the back propagation neural network”, Proceeding of IJCNN, 1989.

[6] M. C. Mozer, R. Wolniewicz, D. B. Grimes, E. Johnson, and H. Kaushansky “Predicting Subscriber Dissatisfaction and Improving Retention in the Wireless Telecommunication Industry”, IEEE Transactions on Neural Networks, 2000.

[7] J. Ferreira, M. Vellasco, M. Pachecco, and C. Barbosa, “Data mining techniques on the evaluation of wireless churn”, ESANN2004 proceedings, 2004.

[8] H. Hwang, T. Jung, and E. Suh, “An LTV model and customer segmentation based on customer value: a case study on the wireless telecommunication industry”, Expert Systems with Applications, 2004.

[9] M. Pethick, Mi. Liddle, P. Werstein, and Z. Huang, “Parallelization of a Backpropagation Neural Network on a Cluster Computer”, 2003.

[10] K. Ganeshamoorthy and D. N. Ranasinghe, “On the Performance of Parallel Neural Network Implementations on Distributed Memory Architectures”, 2008.

[11] S. Suresh, S. N. Omkar, and V. Mani, “Parallel Implementation of Back-Propagation Algorithm in Networks of Workstations”, 2005.

[12] C. T. Chu, S. K. Kim, Y. A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun, “Map-reduce for machine learning on multicore”, Stanford University, NIPS, 2006.

[13] M. T. Hagan, H. B. Demuth, and M. H. Beale, “Neural Network Design”, PWS Publishing Company, 1996.

[14] R. O. Duda, P. E. Hart, and D. G. Stork, “Pattern Classification, Second Edition”, John Wiley, 2001.

[15] Hadoop: http://hadoop.apache.org.