Improved Random Forest Algorithm for Stream Big Data Processing

Jing Li[1], Yingchun Liu[2]
1.College of Computer Science and Technology, Huaqiao University, Xiamen, 361021, China
2. Industrial and Commercial Bank of China, Peony Card Center, Beijing, 100140, China

Abstract: Stream computing is an important form of Big Data computing. Random Forest method is one of the most widely applied classification algorithms at present. From the actual requirements, Random Forest method faces not only huge number of features but also constantly changing data pattern over time. The accuracy of a Random Forest algorithm without self renewal and adaptive algorithm will gradually reduce over time. Aiming at this problem, this paper analyzes the characteristics of Random Forest algorithm, gives a new pruning idea according to the accuracy of the decision trees. In order to adapt to the change of data, a new random method based on margin is presented. This new method can update itself constantly and can be applied in streaming Big Data environments. Using the actual customer data, the new method is verified has higher accuracy in classification.

Keywords: Random forest; Big data; Stream computing

1. Introduction

The importance of Big Data analysis is needless to say in the recent years [1]. In various application scenarios, Big Data computing technologies have two typical processing modes [2-5]: batch computing and stream computing. The batch computing based systems collect and store the data firstly, and then process the stored static data in the centralized way to discover the data value. On the contrary, the stream computing based systems are unable to determine the data sequence or its arrival time, also cannot store all the data that they have received. These systems process the data in the memory in real-time right after the data flow coming, and then output the useful information.

The batch computing technologies for Big Data processing is relatively more mature [6, 7]. The Apache hadoop and MapReduce model of Google have been wildly used in the batch computing based systems [11]. In the higher accuracy and comprehensiveness requirements scenarios, batch computing is the better choice. In the higher real-time but lower accuracy requirements scenarios, if the receiving data amount is unable to be sure before it is received, the stream computing has obvious advantages [8, 9]. Compared to the large number of batch computing technology research, the research works on stream computing technology are relatively small. Early stream computing research works are mainly based on the streaming data calculation in database environment.

Along with the growing requirements for Big Data on Internet, the stream computing technologies based systems that can deal with the real-time, burst and unlimited data flow are appearing, such as the Distributed Stream Computing Platform (S4) [21] from Yahoo!, Storm [20] from Twitter, DFP [22] from Facebook and so on. Each of these systems has its own disadvantages. How to construct high reliability, high throughput, low latency and sustainable operating stream computing based system are the problems to be solved now.

The remainder of this paper is structured as follows. In Section 2, we analyze the features of the stream computing scenarios, discuss the main technical characteristics that the stream computing technologies could have, and introduce the original Random Forest algorithm [10] which is proposed by Breiman etc. in 2001. Section 3 presents an improved Random Forest algorithm based stream computing method that could be applied in Internet Big Data environments. Section 4 describes a series of experiments on a real customer dataset of the financial industry and analyzes the results. Section 5 is the conclusion and future work.

2. Stream Computing and Random Forest

2.1. Stream computing

Generally, big data streaming computing has the following defining characteristics [12].

The input data stream is a real-time data stream and needs real-time computing, and the results must be updated every time the data changes. The data value should be found during very short time, or else the data value will be much lower after quickly;

The rate and order of input stream are indeterminate, and always change very frequent. This requires the system to

have a high throughput that can quickly handle large data volume;

Incoming data arrive continuously at volumes that far exceed the capabilities of individual machines, so most of the data cannot be kept in the machines for long, but deleted right after it coming. The system only has one chance to deal with these data;

As the amount of data in the input stream is infinite, the suspension of service to update Big Data analysis system is not acceptable in real application scenarios. Considering the incremental data, the system should be able to run stably for a long time and self update at any time.

The Internet is a typical streaming Big Data application scenario. A massive amount of data is generated during the daily operation of Internet, which not only includes automatically generated user, user behavior, and log, but also includes all kinds of data which users share to each other in real time. Due to the high requirements of the system response time of the Internet industry, these data often require real-time analysis and calculation in order to provide better services to users.

Here we analyze three typical application scenarios for stream computing over the Internet Big Data.

Social network sites

The social networking sites always perform real-time analyzing on user information. On one hand, they push the information that the user releases, on the other hand they discover and recommend the information to the user which he/she might be interested in, or discover and prevent the fraud, to improve the user experience.

Search engine

In addition to send the search result to end user, better search engines also consider and calculate the user's searching history data to find out his/her interests. Then, they will push some information or advertisements to the user according to the preferences.

Electronic commerce

Electronic commerce focuses on the analysis of user preferences and association analysis by using Big Data technologies, so as to recommend products to user on purpose. Meanwhile, with a lot of e-commerce systems begin to embed Internet consumer financial services, the user's risk analysis and early warning are also very important areas.

With the continuous development of technology, Internet and Internet of things and other areas connect to each other more and more deeply; the amount of data to be analyzed in the future will inevitably be explosive growth. The traditional batch computing model is not suitable for this kind of scenario, which requires a very short response time. But stream computing mode can handle it.

2.2. Random forest algorithm

Random Forest algorithm is a combination classification method proposed by Breiman in 2001. Using bagging method, Random Forest method will draw multiple training sample sets that are different from each other. Every single sample set builds a decision tree with randomly selected attributes.

Random Forest uses CART algorithm for building trees. Considering the large number of built trees, Random Forest method is characterized with good ability to resist noise and outstanding performance in the classification capability [10, 13].

Random Forest is defined as a set of decision trees $H=\{h(x, \theta k),k=1,\cdots\}$, where $h(x, \theta k)$ is a meta-classifier, namely, a un-pruned decision tree created using CART algorithm; x serves as the input vector, while $\{\theta k\}$ is an independent and identically distributed random vector. They determine the growth process of each decision tree. According to the input data, each decision tree will give a result; integration of multiple results will give the final output of a Random Forest.

Each meta-classifier $h \in H$, is equivalent to a mapping function from the input space X to the output class set Y. For each input xi from the input space X, there is $h(x_i)= y_i$, and y_i is the result which is made by the meta-classifier h. If we define a decision function as D, then the final result y as following:

$$y = D\left\{h : x \to \sum_{h \in H} a_h h(x_i) \middle| a_h > 0, \sum_h a_h = 1\right\} \quad (1)$$

In the Random Forest, the growth process of a single decision tree is as the following three steps:

Step 1: For the original training sets, bagging method is used to select random data with replacement and thus form training sets with difference;

Step 2: The features are also selected using sampling approach. If it is assumed that a data set has N features, then M features will be sampled from N, where M<<N. For each extracted training set, only randomly selected M features rather than all N features will be used for node splitting in building trees;

Step 3: All built decision trees will grow freely without pruning.

The final result can be integrated using simple majority voting method (for classification problems) or mathematical average method (for regression problems) performed among the results of the decision trees.

Random Forest method can be deemed as a combining classifier algorithm or a combination of decision trees. It combines the following advantages of bagging and random feature selection [10]:

Bagging can estimate not only the importance of each feature but also generalization error;

The trees of Random Forest method are built using CART algorithm, which is compatible with the treatment of continuous and discrete attributes;

Random Forest method can effectively solve the problem of unbalanced classification;

Random Forest method has excellent noise tolerance and high classification accuracy.

In the context of big data environment, Random Forest method is characterized with following advantages:

From the perspective of the huge amount of big data, the Random Forest method can be competent;

The relatively simple decision trees generated by Random Forest method facilitate business analysts to interpret its meaning;

Random Forest method is suitable for distributed and parallel environment, showing a good scalability;

The simple classifier created by decision trees can process data efficiently, which is applicable to the characteristics of rapid data refresh rate in the big data environment.

Despite of the abovementioned advantages, Random Forest method is also facing new challenges in the mode of big data.

Firstly, in the big data environment, the data update rate is very fast, so are the update rates of data characteristics and modes hidden in data. Decision trees based on training sets data will become out of date and less accurate in classifying data after a certain period of time. This requires algorithms in the big data environment to have data adaptability. Meanwhile, this ability should also be quickly reflected in the classifier, while the normal conduct of business or the stream-form passage of data through the classifier should not be affected.

Secondly, the decision tree, in fact, is a greedy algorithm that easily leads to instability and over fitting. Solving this problem has a great significance for improving the accuracy of decision tree.

Thirdly, the forest scale established by Random Forest has not been clearly defined; oversized scale may result in redundancy and thus reduce the efficiency and accuracy of classification.

3. Improved Random Forest Algorithm

3.1. Related work

In the past, the improvement of the Random Forest method is mainly focused on the following aspects:

In [14], Shilei Liang combined the Random Forest with Hadoop, Map Reduce and other computing frameworks to implement distributed Random Forest and improve the processing efficiency of the algorithm.

In [15, 16], they preprocessed the data to reduce the unbalance of the dataset and promoted the accuracy and classification performance of the algorithm on the unbalanced dataset.

In [15], Zhengfeng Cao replaced the C4.5 with the higher efficiency of node splitting algorithm, such as CHI2, that can improve the ability of the algorithm to process Big Data sets.

Based on the similarity measure and classification interval of classifiers, the redundant classifiers were pruned in order to get better classification effect and smaller forest scale [17, 18, and 19].

These improved methods can effectively improve the performance of the Random Forest algorithm in some certain conditions, but it cannot completely meet all of the requirements in the Big Data environment.

To meet the requirements of stream Big Data computing, this work improves the classical Random Forest algorithm in the following four aspects:

To guarantee that the streaming data can be processed in real-time, the Random Forest algorithm is used to process it. For the Random Forest is a relatively simple classifier, the response time of the data processing can be very short.

Only storing the data for a period of time, and processing a small amount of data under the condition of memory available, that can solve the volatility and the infinite problems of the streaming Big Data.

Due to the order of input data stream is indeterminate, the classical Random Forest classifiers are unable to deal with all the input data. Our new classifiers should be updated when the new data coming and keep their sensitivity and accuracy to the data. Because of the volatile of the data, the updating of the classifiers must be based on the limited training data stored temporarily.

The classifier updating method must be scalable and efficient, and not affect the normal processing of the data.

3.2. Improved random forest algorithm

Before improving the algorithm, the accuracy Ah of the decision tree h from the Random Forest is defined as following:

$$A_h = \frac{n_r}{n} \tag{2}$$

In the formula (1), n_r is the number of times the decision tree h gives the correct result. n is all the number of data processed times gives by the decision tree h. Accuracy shows the ratio of the correct results given by a tree in a certain period of time.

For the classification problem, it is considered that the decision tree gives a correct result if the result given by decision tree h is consistent with the final classification result. For the regression problem, it is required to calculate the difference between the result xi given by decision tree h and the final result , and their standard deviation will be taken as the accuracy rate of h:

$$s = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - m)^2} \tag{3}$$

$$A_h = s \tag{4}$$

According to the accuracy rate , we can only measure the accuracy of a tree h in period of time.

The idea of algorithm improvement is to track the accuracy rate of each tree during the implementation process, and regularly update the forest and eliminate those trees with lowest accuracy rate. The improved Random Forest is as follows:

Constructing a decision tree group H in accordance with classical Random Forest algorithm;

Building a record sheet T_h for each decision tree h∈H to record the generated results during execution;

After running for some time, the record sheets of all decision trees are scanned to pick out and delete those trees with lowest accuracy.

After accuracy screening, the number of trees in the forest will be reduced, thus realizing the pruning of all decision trees. However, excessive reduction in the number will also lead to the reduced accuracy in the whole decision tree sets [11].

In order to keep an appropriate number of decision trees and maintain the quality of the forest, we track the dataset while pruning to generate new decision trees. To screen out more useful samples for decision trees from the datasets, we introduce the definition of margin.

Margin refers to the overall decision-making correctness rate of a Random Forest on a piece of given sample data (x, y). It is calculated as in the formula (5).

$$m\arg in(x,y) = av_k I(h_k(x) = y) - max_{j \neq y} av_k I(h_k(x) = j) \quad (5)$$

Here, av_k is an averaging function and I is a metric function. If the correct results with respect to sample (x, y) can be obtained from most decision trees in the Random Forest, then margin(x, y) is bigger than zero. The case of margin(x, y) smaller than zero indicates that the sample is wrongly identified by most decision trees, suggesting that the algorithm draws a wrong conclusion on the sample.

Samples with margin(x, y) bigger than zero illustrates that decision trees are able to gain right results. Since those decision trees in high similarity with the existing trees will not improve the accuracy of entire forest, such samples do not need to be processed again.

Samples with margin(x, y) smaller than zero will be recorded and used to form a new training dataset S' so as to allow the newly generated decision trees to improve the accuracy of entire forest. Although the dataset S' only account for a small part in the entire dataset S, its data features are very different from other data.

By applying the Random Forest algorithm on the data set S', a new decision tree set {h'(x, θ k), k=1,…} is thus obtained. Since the dataset S' represents only a small part of data from the entire dataset, a certain proportion of decision trees should be screened from this set and added to the original decision tree set.

The number of decision trees to be screened can be determined based on the ratio between the data set S' and the entire data set S. The details are presented in the formula (6).

$$N_{new} = N_{all} \times \frac{number(s')}{number(s) - number(s')} \quad (6)$$

The screening methods may involve those as follows:

Method 1, based on the accuracy sorting obtained by testing the datasets S', N_{new} decision trees with maximum accuracy will be selected.

Method 2, based on the accuracy sorting obtained by testing the entire datasets S, N_{new} trees with maximum accuracy will be selected.

Method 3, calculating the ratio between margin mean and margin variance of each tree on the data set S'[18], which is taken as the importance measurement index for each tree, N_{new} trees with highest importance will be selected.

The improved Random Forest algorithm is in Figure 1:

Generate initial Random Forest H by using initial training data set S;

Classify the current data set Si to be processed using Random Forest H;

Classify each x_j in Si with each tree h_j in the Random Forest;

Record the classification result of each tree and each data, and calculate the interval value margin(x_j , y) of the data classification result;

Add x_j to the new training data set S' if margin(x_j , y) is less than the given threshold.

Calculate the accuracy of each tree and prune the tree after the classification of Si is completed;

Perform the Random Forest algorithm on the new training data set S' to generate a new Random Forest H';

Prune the new Random Forest, then combine H' and H after pruning to form the new Random Forest H;

Clear the training data set S', and start to process the next batch of data.

3.3. Advantages of the new random forest algorithm

The new Random Forest algorithm has the following advantages:

The new algorithm has a limited data set each time. In practical applications, the size of the data set can be designed according to the memory size to ensure the real-time calculation and computational efficiency of the data;

In the new algorithm, only the result record table and the new training data set need to be stored, which is much smaller than the original data stream. It meets the volatile characteristics of streaming big data, and has better scalability under large data volume;

The processing of new data only needs to use Random Forest for verification and voting. It increases the execution efficiency, so the processing result of the data can be fed back in real time;

This system can continuously update itself with the new features of the data to meet the disorder and infinity of the streaming big data environment.

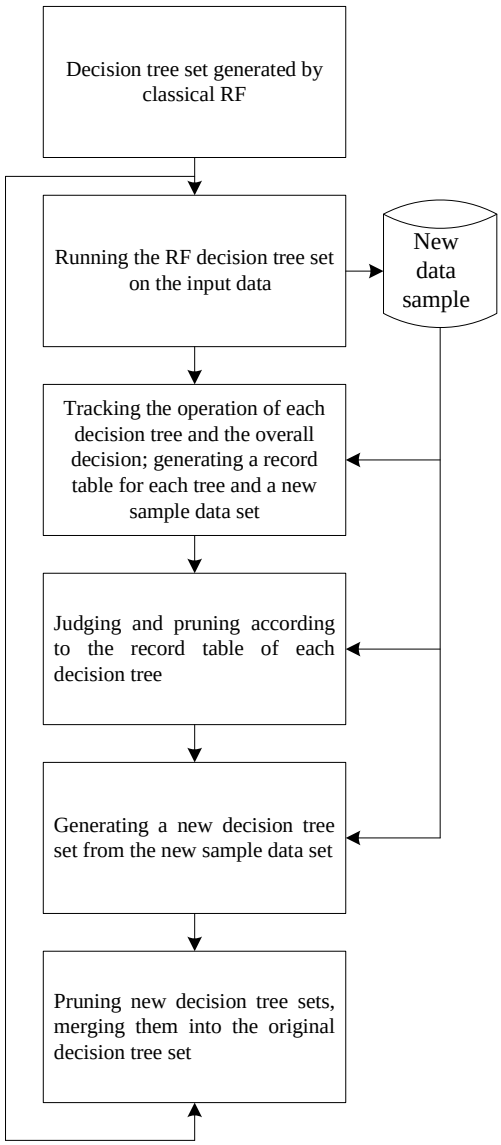

Figure 1. Improved random forest algorithm

4. Experiments and Results

The dataset, we call "Bank", which is used in the experiments are originated from the real credit cardholders of financial industry. The data amount accounts for 2,000,000 pieces with around 100,000 pieces of data from each quarter, which was sampled from a larger original 5-year dataset. The dataset contains a target category and 15 features attributes, which includes both the continuous numerical attributes and discrete attributes. The details of the features are showed in table 1.

Custom value is the target category which can be evaluated from 1 to 6. The higher value means the high credit of the customer. In the experiments, we evaluated the custom value by considering 15 features, which include age, gender, marriage, city, job, title, education, salary, yearly income, family size, child number, house type, purchases, activity and preference.

Random Forest kit in R language version was used in the test. In order to verify the effectiveness of abovementioned improvements, we used 100,000 pieces of data of the first quarter in the first-year as the initialized training

set and use them to establish the initial Random Forest, where the number of trees equals 100.

We establish the Random Forest to use multiple classifiers to evaluate the customer, so we can offer better performance on the incremental improvements and better adapt to the changes in the data than the original single classifier. The method of using Random Forest multiple classifiers for customer evaluation is illustrated in Figure 2. Each single customer's data is distributed to 100 decision trees to be deal with. Each decision tree outputs a customer evaluation result. And the final evaluation of each customer is generated by majority voting method.

Table 1. The features of real customer data

Target attribute	
Name	Value
Custom Value	1 / 2/ 3/ 4/ 5/ 6
Feature attribute	
Name	Value
Age	0 - 100
Gender	Male / Female
Marriage	Y / N / D
City	Beijing, Shanghai....
Job	Farmer, Teacher....
Title	Manager, VP....
Education	High school, College....
Salary	0 - 1000000+
Yearly Income	0 - 10000000+
Family Size	0 - 10
Child Number	0 - 10
House Type	Own, Rent....
Purchases	0 - 1000000+
Activity	0 - 1000000+
Preference	0 - 1000000+

Firstly, we validate when data changes with the time, whether its pattern will change and if such changes will affect the accuracy of the algorithm. Through taking 100,000 pieces of data in the first quarter of the first year as initialized training set to establish Random Forest and using the established Random Forest to classify the data of each following quarter, it was clearly observed that the original Random Forest gradually became unadapted to new data and its accuracy also reduced slowly with the change of time. The accuracy is from 93% to 81% during five years. We set the pruning rate as 50%, use the improved Random Forest algorithm to generate new decision trees, and then add these trees into the original Random Forest. By using the three decision tree screening methods in last section, the evaluation accuracy of the improved Random Forest algorithms has been shown in Figure 3.

It is obviously that comparing to the original Random Forest algorithm, the improved algorithms adapt to the change of the data model in a better way. Because of the whole forest changing with the time, the accuracy of the latter algorithms can keep above 90% all the time. Among those methods for screening new decision trees, both the screening method by using the accuracy rate of new sample data set S' (method 1) and the screening method by using the ratio between each tree's margin mean and margin variance (method 3) can gain satisfactory results.

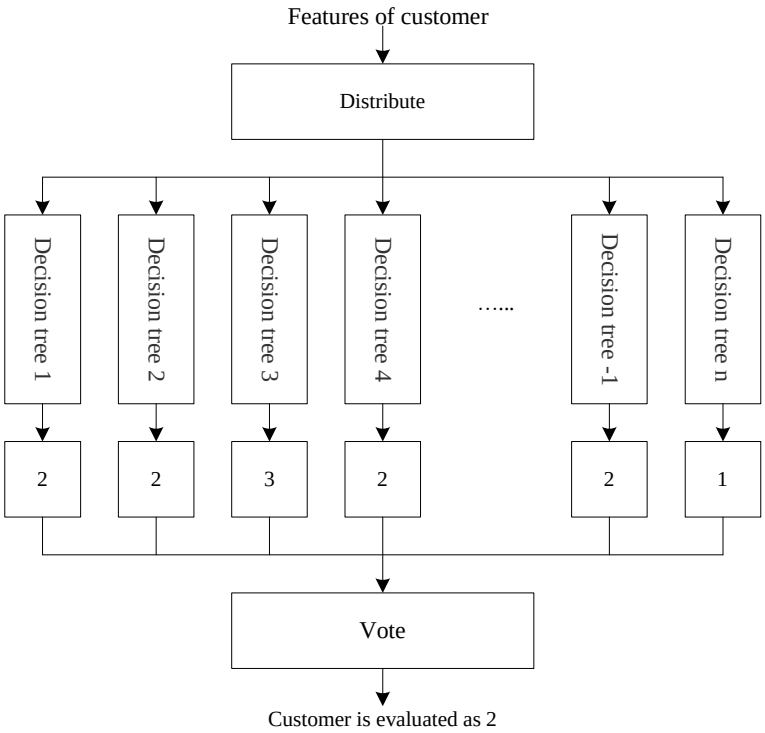

Figure 2. Using random forrest to evaluate customer

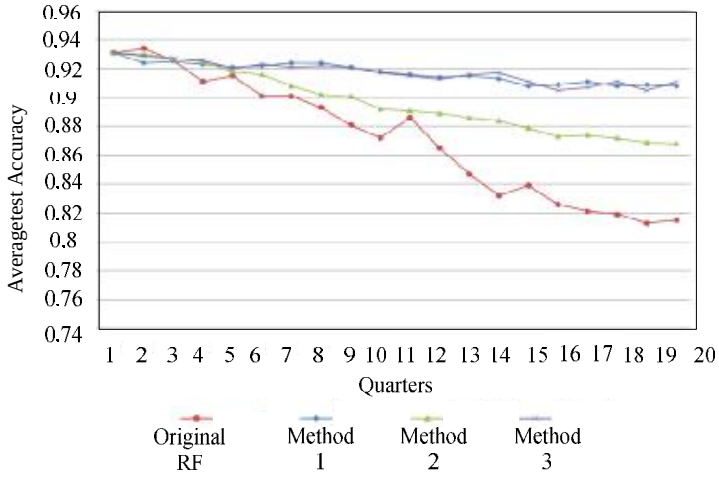

Figure 3. Accuracy comparison on the improved random forest method

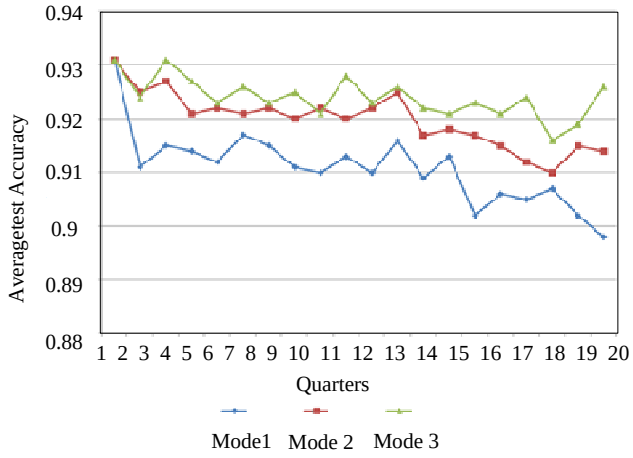

Figure 4. Accuracy comparison on three modes

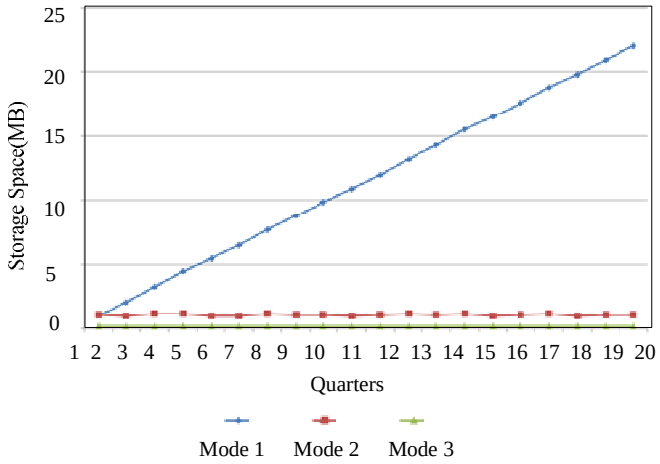

Figure 5. Storage space comparison on three modes

Another problem we have considered is that how could the training dataset of the new forest affect the accuracy of the method. If the memory limitation is ignored, we use the following three modes to update the Random Forest after the end of each quarter.

Mode 1: using all the data from the beginning to now to retrain the Random Forest;

Mode 2: using the data from the last quarter to retrain the Random Forest;

Mode 3: using the improved Random Forest algorithm to retrain the Random Forest.

The experiment results of these three modes are presented in Figure 4. The accuracy of mode 3 is the best, and mode 2 is in the second place. The accuracy of mode 1 decreases a lot when the amount of data is increasing.

In Figure 5, we give the storage space required for each of these three modes. Model 1 needs the storage space to increase linearly with the time and the quantity of data. It consumes the most space in these three. The storage space that mode 2 requires is related to the total amount of data in each cycle. When the data flow changes slowly, it will remain stable. It needs less space than the Mode 1 does. Mode 3 needs the least space which is much smaller than the model 2 needs. And its volatility is only related to the current accuracy of the algorithm, but not to the total amount of data.

This new algorithm can work normally in the case of the customer data only goes through the classifier once. There is no need to store or scan the massive historical data. So, it requires little storage space. It does self adjustment according to the transformation of the stream Big Data to adapt to the newest data flow. This adaptive method ensures the data throughput and processing efficiency while maintaining the accuracy of the data processing. It works well on the dataset "Bank".

5. Conclusion and Future Work

This work improved the original Random Forest algorithm, proposed a new method IRF (Improved Random Forest), which could adapt to the stream Big Data processing. IRF just needs the stream Big Data go though the classifier instead of storing and scanning all the massive historical data. It requests for very low storage space. It does self adjustment according to the transformation of the stream Big Data to adapt to the newest data flow. This adaptive method ensures the data throughput and processing efficiency while maintaining the accuracy of the data processing. The results of the experiments on real stream Big Data from the financial industry show that IRF offers good performance, which means that our attempts can solve the real problems encountered in the stream Big Data processing scenarios on Internet. Especially, its data patterns will also change gradually with time, which allows it to give a better play in the Big Data scenarios.

There are still plenty of directions in this area which are worth to explore. For example, does IRF have good performance on other types of datasets? Could the efficiency of the pruning decision function be improved? What proportion of the new decision tree should be appropriate? Besides, how to combine the improved Random Forest algorithm with Storm, S4 and other distributed large data processing architecture to achieve better system fault tolerance, resource scheduling, load balancing performance also need to be further studied.

6. Acknowledgments

Acknowledge the support of the Natural Science Foundation of Fujian Province, China (Grant No. 2015J01258) Scientific Research Funds of Huaqiao University, China (Grant No. Z14Y0036)

References

[1] Jiawei Han, Kamber.M. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers. March, 2006.

[2] Meng XF, Ci X. Big Data management: Concepts, techniques and challenges. Journal of Computer Research and Development. 2013, 50(1), 146-169.

[3] Lim L, Misra A, Mo TL. Adaptive data acquisition strategies for energy-efficient, smartphone-based, continuous processing of sensor streams. Distributed and Parallel Databases. 2013, 31(2), 321-351.

[4] Li B. D., Mazur E, Diao Y. L.. SCALLA: A platform for scalable one-pass analytics using MapReduce. ACM Trans. on Database Systems. 2012, 37(4), 1-43.

[5] Yang D., Rundensteiner E.A., Ward M. Mining neighbor-based patterns in data streams. Information Systems. 2013, 38 (3), 331-350.

[6] Li G.J., Cheng X.Q.. Research status and scientific thinking of Big Data. Bulletin of Chinese Academy of Sciences. 2012, 27 (6), 647-657.

[7] Wang Y. Z., Jin X. L., Cheng X. Q.. Network Big Data: Present and future. Chinese Journal of Computers. 2013, 36(6), 1125-1138.

[8] Kobielus A. The role of stream computing in Big Data architectures. 2013. http://ibmdatamag.com/2013/01/ the-role-of-stream-computing-in-big-data-architectures/.

[9] Sun DaWei, Zhang GuangYan, Zheng WeiMin. Big Data Stream Computing: Technologies and Instances. Ruan Jian Xue Bao/ Journal of Software. 2014, (4), 839-862.

[10] Breiman L. Random Forests. Machine Learning. 2001, 45(1), 5-32.

[11] Qin X. P., Wang H. J., Du X. Y., Wang S. Big Data analysis: Competition and symbiosis of RDBMS and MapReduce. Ruan Jian Xue Bao/ Journal of Software. 2012, 23(1), 32-45.

[12] Qian Z. P., He Y., Su C. Z., et al. TimeStream: Reliable stream computation in the cloud, Proc. 8th ACM European Conference on Computer Systems, EuroSys 2013, Prague, Czech republic, ACM Press. Apr, 2013, 1-14.

[13] Xuechan Li. Research on Classification Calculation Way of a Great Amount of Data According to the Database Sampling. Computer Science. 2008, 35(6), 299.

[14] Shilei Liang. Research and implementation of image classification system based on Hadoop platform. Xiamen University. Master thesis, 2014.

[15] Zhengfeng Cao. Study on Optimization of Random Forests Algorithm. Capital University of Economics and Business, Doctoral dissertation. 2014.

[16] H.P. Zhang, M.H. Wang. Search for the smallest Random Forest, Stat. Interface 2009, Vol2, p381-388.

[17] M. Robnik-Sikonja. Improving Random Forests, Proceedings of the 15th European Conference on Machine Learning. 2004, p359-370.

[18] Leistner C, Saffari A, Santner J, et al. Semi-supervised Random Forests. IEEE 12th International Conference on Computer Vision, IEEE. 2009, 506-513.

[19] Chunhua Shen and Hanxi Li. Boosting through Optimization of margin Distributions. IEEE Transactions on Neural Networks . 2010, Vol 21, No 4. P659-666.

[20] Storm. 2013. http://storm-project.net/.

[21] Neumeyer L, Robbins B, Nair A, Kesari A. S4: Distributed stream computing platform. In: Proc. of the 10th IEEE Int'l Conf. on Data Mining Workshops (ICDMW 2010). Sydney: IEEE Press, 2010. 2010, 170-177.

[22] Borthakur D, Sarma JS, Gray J, Muthukkaruppan K, Spigeglberg N, Kuang HR, Ranganathan K, Molkov D, Mennon A, Rash S, Schmidt R, Aiyer A. Apache hadoop goes realtime at Facebook. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2011 and PODS 2011). Athens: ACM Press. 2011, 1071-1080.