CHEMICAL ENGINEERING TRANSACTIONS

VOL. 51, 2016

A publication of AIDIC
The Italian Association of Chemical Engineering
Online at www.aidic.it/cet

Guest Editors: Tichun Wang, Hongyang Zhang, Lei Tian
Copyright © 2016, AIDIC Servizi S.r.l.,
ISBN 978-88-95608-43-3; **ISSN** 2283-9216

DOI: 10.3303/CET1651065

The Application of high-dimensional Data Classification by Random Forest based on Hadoop Cloud Computing Platform

Chong Li

Information Engineering School, Chongqing Vocational Institute of Engineering, Jiangjin, Chongqing, China, 402260
cqlichong@163.com

The high-dimensional data has a number of uncertain factors, such as sparse features, repeated features and computational complexity. The random forest algorithm is a ensemble classifier method, and composed of numerous weak classifiers. It can overcome a number of practical problems, such as the small sample size, over-learning, nonlinearity, the curse of dimensionality and local minima, and it has a good application prospect in the field of high-dimensional data classification. In order to improve the classification accuracy and computational efficiency, a neval classification method based on the Hadoop cloud computing platform is proposed. Firstly, the processing of Bagging algorithm is done with the data sets to get the different data subsets. Secondly, the Random Forest is completed by training of the decision tree under the MapReuce architecture. Finally, the processing of data sets classification is done by the Random Forest. In our experiment, the three high-dimensional data sets are used as the subjects. The experimental results show that the classification accuracy of proposed method is higher than that of stand-alone Random Forest, and the computational efficiency is improved significantly.

1. Introduction

The ensemble classifier methods have been developing rapidly in decades, since the performance is always superior to that of other single classifier methods(zhang, 2010; Sortiris, 2011). And it has become one of the main domains in machine learning and pattern recognition. The main process of constructing an ensemble classifiers system is to exploit base classifiers with different subsets of the same training data set, and then combines the outpus of these variant base classifiers (Valiant et al., 1984). In the research of ensemble classifiers method, the accuracy of base classifier and the diversity between different base classifiers are two main factors. It leads to a better classification performance with higher individual accuracy and higher diversity in ensemble classifiers system. Among the ensemble learning methods, the most popular ensemble algorithms are Bagging (Breiman, 1996), AdaBoost(Freund et al., 1995), Random Subspace(Ho et al., 1998)), Random Forest (Breiman, 2001) , and Rotation Forest(Juan et al., 2006)).
In research of ensemble classifiers method for high-dimensional data classification task, there are some more problems to deal with. The common factors of ensemble classifiers method are as follows, the accuracy of base classifier, the diversity between each two base classifiers, the number of base classifiers to be combined, the strategy of combing the base classifiers and so on. In addition to these, there are some other problems for high-dimensional data classification. 1) The high dimension of data. In our research, the dimensions of data sets in our experiments are 250 to 10000. The result shows that the accuracy of base classifers and the diversity between them can decrease with the increase of dimension. 2) Computational complexity. As we know that with the increase of data's dimension, the computational complexity increases. So, the solutions to decline the computational complexity are feature extraction and less base classifiers to be combined.
Random Forest is a very popular ensemble classifier method, the main idea is to construct a strong classifier by combining multiple weak classifiers. However, when dealing with the high-dimensional data classification, the performance degrades rapidly. According to the Hadoop cloud computing platform, the Random Forest algorithm is deployed on the Hadoop platform to realize the parallel computing, which can increase the computational speed in theory. Random Forest on Hadoop platform improves both the computational

efficiency and accuracy of the classification of high dimensional data. MapReduce architecture is one of the most classical Hadoop platform, and is an embodiment of "divide and rule". In the Map stage, it can decompose a complex problem into a number of sub problems that are easy to solve, and on the basis of solving the sub problems, the original complex problems is solved esasily. In the Reduce stage, the Random Forest is got to do the data classification by combining the decision trees. Finally, the experiment is done with 3 data sets from UCI, and the results show that the Random Forest on Hadoop platform gets higher classification accuracy than stand-alone Random Forest. Furthermore, it indicates that the computational speed almost increases by 40% than stand-alone Random Forest.

2. Hadoop

Hadoop is a distributed system infrastructure, which is developed by the Apache Foundation (Grangier D.(2008)). The user can build a distributed computing platform without knowing the details of the distributed bottom layer. The difference between Hadoop and other parallel programming methods is that Hadoop has a distributed file system HDFS to support its data processing. In the calculation, each task processing can be stored in the local data file to avoid the transmission of data between nodes. The basic architecture model of Hadoop is composed of the distributed file system (HDFS) and MapReduce computing framework.

2.1 HDFS

HDFS is a distributed storage system of Hadoop platform with high fault tolerance, and it can be deployed on low-cost hardware devices (YANG et al., 2012; ALHAM et al., 2011)). HDFS is suitable for applications of big data sets, and provides a high throughput of data. HDFS is a slave / Master structure, which consists of a name node and a number of data nodes. And name node is used as a server to save the metadata in the file system, so as to manage the namespace of file system and the access to files by clients. The name node does not store the actual data file, because the data files are divided into a number of file blocks according to the rules to be stored on different data nodes.

2.2 MapReduce

MapReduce is a parallel computing framework of Hadoop platform, and Its inspiration comes from the functional programming language Map and Reduce (Dean et al., 2008; Rranger et al., 2007). In dealing with some chaotic data, the feature of each data is extracted by the analysis of Map function, and the <*key, value*> is got as the input. In Shuttle stage, all values that belong to the same key are put into the same Reduce function for processing. In general, the MapReduce framework is divided into Map tasks and Reduce tasks. Multiple data segments are processed by using Map function on multiple machines, and the local results obtained are combined with the combiner function. And then the data is distributed to different Reduce functions to be processed with <*key, value*>. When all the Map and Reduce jobs are completed, a number of output files of Reduce function are put into another MapReduce program to do the combination processing. Then the result is wrote on the HDFS. MapReduce programming model provides a highly abstract programming mechanism, the programmers only need to know what to do and how to program. Because data synchronization, data distribution, data communications and other details are given to the underlying framework of the system to deal with, so as to facilitate the programmer to consider how to design algorithms from the nature of the problem.
In the MapReduce platform, the data set of a job is usually divided into several independent databases. And a <*key, value*> pair is mapped into a new <*key, value*> pair by Map parallel processing, which is as the input of Reduce. The calculation process of MapReduce is shown in Figure 1, which includes input stage, Map stage, Reduce stage and output stage.

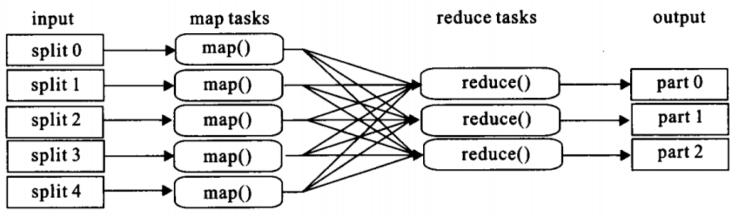

Figure 1: The flow chart of MapReduce

3. Random Forest

Random forest is a new ensemble classifier proposed by Breiman in 2001. It adopts the classification and regression tree (CART) as a weak classifier, and the Bagging method is used to generate different training sets (Breiman, 1984). In the construction of a single decision tree, random selection of the features of the internal nodes is used to split the properties of the data set. The combination of the Bagging method, the CART algorithm, and the random selection of features makes Random Forest can tolerate noise better and get better classification performance.

3.1 Decision Tree

The base classifier of Random forest is decision tree, the data structure of decision tree is used to deal with business logic. The leaf node in the decision tree is the result of classification, and non-leaf node is the decision node (Quinlan, 1986). The classification of decision tree is divided into three parts. Firstly, a decision tree is constructed by determining the structure of the training data set on each non-leaf node. Then, the path of the root node to the leaf node is analysed to find the classification rules. Finally, the test data set is classified by these rules. Decision tree only outputs a classification result for a data, because there is only one path from the root node to any leaf node. In general, the decision tree is regarded as a single classifier.

3.2 Bagging

Bagging algorithm is proposed by Breiman in 1996, it generates different component classifiers by different training samples (Breiman, 1996). Bagging algorithm is based on bootstrap sampling method, which means that the training sample set T_{set} is randomly selected from the original sample set S, and the number of sample set T_{set} is the same as that of the original sample set S.

Assuming that N is the number of samples in the original sample set S , when the Bootstrap method is used to sample from the set S, the probability of each sample in S that is not extracted is $(1-1/N)^N$. If N is very large, $(1-1/N)^N \approx 0.368$, it means that nearly 37% samples of the original set S does not appear in the T_{set}. By constructing different training sets, the difference between the classifiers is increased, that is the reason why Bagging method can improve the generalization ability of the combined classifiers. Through the T round of training, a classifiers sequence $\{c_1, c_2, ..., c_T\}$ is got. The final classification results are obtained by the simple majority voting method or simple averaging method with these classifiers sequence.

3.2 The implementation of Random Forest

Random forest is based on K decision tree classification model $\{h(x, \theta_k), k=1,2,\ldots,T\}$, where, x is the inputs, θ_k is independent and identically distributed random vectors. For each input x, each decision tree has one vote. The classification idea of random forest classification is as follows. Assuming that the training sample set D including N_D samples, and a new N_D samples set are selected by Boostrap method from set D. Then, F features are selected from features of T_{set} to construct the decision trees, so T decision tree classifiers are generated. Finally, the decision tree is combined to form a random forest. The key steps of the random forest algorithm are as follows

1) Get data training sets. N_D samples are selected by Boostrap method from D, and the iteration of the above repeats T times, this method is called Bagging. Then data training set $D_i(i=1,2,\ldots,T)$ is got.

2) construct decision tree. k features are selected from K features of T_{set} to construct the decision trees, where $k<<K$, in general, k is set as log(K+1). Each training sample subsets with k features is used to constructed a decision tree, and T decision trees are got by T times to generate Random Forest.

3) Voting decision. The formula of majority voting method is as follows.

$$H(x) = \arg\max_{Y} \sum_{i=1}^{T} I(h_i(x) = Y) \tag{1}$$

Where, $H(x)$ is the Random forest model, h_i is the decision tree model, Y is the label of testing sample x, and $I(\cdot)$ is a indication function.

4. High-dimensional data classification by Random Forest base on Hadoop platform

When dealing with high dimensional data sets, the stand-alone Random Forest algorithm spends a lot of time, and the computational efficiency is low. In this paper, a new method is proposed, which is the Random Forest algorithm based on Hadoop cloud computing platform.

In the implementation process, firstly, the random sampling of the original data set by through Bagging algorithm is done to construct sample subsets for each decision tree. Bagging algorithm is the extraction of the back, and the sampling process is independent between each other in each time, so it does not interfere with each other. Samples selected from samples sets are repeated several times, and some samples may not appear at all. So, it makes that the algorithm has a good generalization ability and stability. After the completion of the construction of the sample subsets, the construction of decision tree is done on the Map stage. The construction process of each decision tree corresponds to a Map task, and the number of decision trees determines the number of Map tasks. So, the number of decision tree to be constructed is reasonable according to the computing capability of the cluster, and node splitting is the most important part of the construction process. In Reduce stage, The trained decision tree model is transformed to HDFS.

In the process of constructing a decision tree, the splitting attribute of each node is needed. This process needs to traverse all the attributes of the candidate attributes, and calculate the corresponding information gain of each feature. So, this process can be further parallelization. In this paper, it is assigned to each child thread to complete the process of calculating the information gain with traversing the candidate attribute set by multi thread method. And then the results of the child thread are used to obtain the optimal partition properties. The specific operation steps are as follows.

1) On the basis of the original sample sets, random sampling by the Bagging algorithm is used to build a sample subset for each tree.

2) According to the sample subset obtained in the first step, the Map task is initialized, and the number of Map tasks depends on the number of trees in a random forest. Then, the Map task is assigned to each of the machines in the cluster.

3) The feature subspace is randomly extracted from feature space to be as the candidate feature subspace in each child thread processing.

4) In each child thread, the information gain of all possible attribute values on its corresponding candidate feature subspace are computed, and the minimum information gain value and the corresponding feature name are returned

5) Comparing the results returned by each child thread, the optimal splitting value and the splitting point of the node are selected.

6) Repeat step 2) to step 5) until iteration ending condition is satisfied. Then the construction of decision tree is completed.

7) When all the Map tasks are done, the construction of all decision tree is completed. Then the Reduce function is executed, so that each decision tree is written on the HDFS to get the final Random Forest classifier.

5. Experiments and analysis

5.1 Introduction of data sets

Three high-dimensional real-world data sets from UCI Repository are applied in our experiment. These data sets include Arcene data set, Gisette data set, and Internet Advertisements (InAd) data set,. Table 1 shows the main characteristics of the these three benchmark classification data sets. The first column gives the name of each data set. The next three columns, respectively, show instance size, attributes size and number of classes of each data set.

Table 1 The characteristics of the used data sets

Data set	Instances	Attributes	Classes
InAd	2359	1559	2
Gisette	1000	5000	2
Arcene	200	10000	2

5. 2 Results and analysis

In this experiment, five computers are used, which are installed with Linux Ubuntu 15.04, and the version of Hadoop is 2.5.0. The first computer is used as NameNode Master service node, and the other four computers are used as DataNode Slave service nodes, each node has the same configuration.

In this experiment, two evaluating indicators are introduced, which are the classification accuracy and the computational speed ratio. In oder to eliminate the system error, three different training sets are obtained from the data sets, and the proportion of the original data set is 30%, 40% and 50%. So, three results are got in each data set by one method.

1) Classification accuracy. Here, recognition accuracy is used as a statistical measure of the binary classification test correctly. We suppose that N_{fp} and N_{fp} are respectively as true and false values of the number of positive samples tested, N_{fn} and N_{tn} are separately as true and false values of the number of negative samples tested. Thus, the accuracy is the proportion of true results (both true positives N_{tp} and true negative samples N_{tn}) in the population, and the accuracy is defined as follows.

$$accuracy = \frac{N_{tp} + N_{tn}}{N_{tp} + N_{fp} + N_{fn} + N_{tn}} \tag{2}$$

Table 2 The classification accuracy by stand-alone Random Forest (%)

Data set	Accuracy		
Proportion of training samples	30%	40%	50%
InAd	91.56	97.76	98.91
Gisette	86.23	91.15	95.32
Arcene	75.01	79.90	86.75

Table 3 The classification accuracy by Random Forest on Hadoop platform (%)

Data set	Accuracy		
Proportion of training samples	30%	40%	50%
InAd	92.16	97.96	99.26
Gisette	88.13	92.41	96.08
Arcene	75.69	81.36	88.34

The classification accuracies of three data sets by two different methods including stand-alone Random Forest and Random Forest on Hadoop platform are shown in talbe 2 and table 3. From the results in one table, we find that with the increase of training samples, the classification accuracy increases in direct proportion. Comparing the results from table 2 and table 3, the conclusion is that the classification accuracy of Random Forest on Hadoop platform is a little higer than that of the stand-alone Random Forst. So, the Random Forest on Hapdoop platform almost has the better classification performance than stand-alone Random Forest.
2) Computational speed ratio. The speed ratio is used to compare the computational time between the stand-alone Random Forest and the Random Forest based on MapReduce cloud computing platform. The formula of speed ratio is defined as follows.

$$S_p = \frac{T_p}{T_s} \tag{3}$$

Where, T_s is the computational time of stand-alone Random Forest, T_p is the computational time of Random Forest on the Hadoop platform. If S_p>1, it means the Random Forest on the Hadoop platform is more effective than stand-alone Random Forest.
Talbe 4 shows the computational speed ratio between Random Forest on Hadoop platform and stand-alone Random Forest, the results are grater than 1 and the numerical value is stable. From all of the results, we can find that the computational speed is increased about by 40%, it is very important in the classification of high-dimensional data.

Table 4 The speed ratio between Random Forest on Hadoop platform and stand-alone Random Forest

Data set	Computational speed ratio		
Proportion of training samples	30%	40%	50%
InAd	1.34	1.35	1.36
Gisette	1.41	1.42	1.39
Arcene	1.43	1.45	1.40

6. Conclusion

In this paper, firstly, the Hadoop cloud computing platform is introduced, which includes distributed file system (HDFS) and MapReduce framework. Then, we state that Random Forest is a very popular method for data classification. Its main idea is to combine several weak classifiers to enhance the performance of the classifier that is called strong classifier. According to the Hadoop cloud computing platform, the parallelization of Random Forest algorithm has been studied. With the parallelization of Random Forest algorithm on Hadoop platform, the computational effciency and classification accuracy of the proposed algorithm are improved. Finally, the experiments are done with three data sets from UCI. The experimental results show that proposed method can improve the speed and perfomlance than the stand-alone Random Forest.

References

Alham N.K., Li M.Z., Liu Y., 2011, A MapReduce·based distributed SVM algorithm of automatic image annotation [J]. Computers and Mathematics with Applications, 62(7): 2801-2811.

Breiman L., 1996, Bagging Predictors [J]. Machine Learning, 24(2): 123-140.

Breiman L., 2001, Random forests [J].Machine Learning, 45(1): 5-32.

Breiman L., Friedman J., Olshen R., Stone C., 1984, Classification and Regression Trees [J]. Belmont, Califomia: Wadsworth International.

Dean J., Ghemawat S., 2008, MapReduce: simplified data processing on large clusters [J]. Communications of the ACM, 51(1): 107-113.

Freund Y., Schapire R.E., 1995, A decision-theoretic generalization of on-line learning and an application to boosting [J]. In: Proceedings of the 2nd European Conference on Computational Leaming Theory, Barcelona, Spain, 23-37.

Grangier D., Bengio S., 2008, A discriminative kernel-based approach to rank images from text alleries [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008, 30(8): 1371-1384.

Ho T.K., 1998, The Random Subspace Method for Constructing Decision Forests [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 8: 832-844.

Juan J., Ludmila I., 2006, Rotation Forest: A New Classifier Ensemble Method [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(10): 1619-1630.

Liu Y., Li MZ., Alham N.K., 2012, HSim: A MapReduce simulator in enabling cloud computing [EB/OL]. http://www.Sciencedirect.com/science/article/pii/S0167739X110.00884.

Quinlan J.R., 1986, Induction of Decision Trees[J]. Machine Leaming, 1: 81-106.

Ranger C., Raghuraman R., Penmetsa A., 2007, Evaluating MapReduce for multi-core and multiprocessor systems [C]. High Performance Computer Architecture. Stanford Univ. 13-24.

Sotiris K., 2011, Combining bagging, boosting, rotation forest and random subspace methods [J]. Aritificial Intelligence Review, 35: 223-240.

Valiant L.G., 1984, A theory of learnable [J]. Artificial Intelligence and Language Processing, 27(11): 1134-1142.

Zhang C.X, 2010, Zhang JS. A variant of Rotation Forest for constructing ensemble classifiers [J]. Pattern Analysis Application, 13: 59-77.