Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 14, no. 5, May 2018, pp. 955-964
DOI: 10.23940/ijpe.18.05.p14.955964

Spark-based Ensemble Learning for Imbalanced Data Classification

Jiaman Ding*, Sichen Wang, Lianyin Jia, Jinguo You, and Ying Jiang

Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, 650500, China

Abstract

With the rapid expansion of Big Data in all science and engineering domains, imbalanced data classification become a more acute problem in various real-world datasets. It is notably difficult to develop an efficient model by using mechanically the current data mining and machine learning algorithms. In this paper, we propose a Spark-based Ensemble Learning for imbalanced data classification approach (SELidc in short). The key point of SELidc lies in preprocessing to balance the imbalanced datasets, and to improve the performance and reduce fitting for the big and imbalanced data by building distributed ensemble learning algorithm. So, SELidc firstly converts the original imbalanced dataset into resilient distributed datasets. Next, in the sampling process, it samples by comprehensive weight, which is obtained in accordance with the weight of each class in majority class and the number of minority class samples. After that, it trains several classifiers with random forest in Spark environment by the correlation feature selection means. Experiments on publicly available UCI datasets and other datasets demonstrate that SELidc achieves more prominent results than other related approaches across various evaluation metrics, it makes full use of the efficient computing power of Spark distributed platform in training the massive data.

Keywords: imbalanced data classification; ensemble learning; comprehensive weight; random forest; Spark

(Submitted on February 4, 2018; Revised on March 5, 2018; Accepted on April 25, 2018)

1. Introduction

Many real-world areas such as telecommunications, health care, biomedicine or financial businesses generate massive amounts of available data which accumulating at an unprecedented rate. The value of data in various fields is more important than ever. How to efficiently and accurately excavate valuable information from data information has attracted wide attention. However, the existing data is characterized by diversity and complexity, and the data distribution is seriously imbalanced. There are many large-scale imbalanced data classification problems, such as text classification [5], disease diagnosis [18] and image classification [9]. However, the classification accuracy of minority class samples is so low that when the traditional classification algorithm is used to classify imbalanced data, there is a decrease in the overall performance [14]. Therefore, it is significant to choose the appropriate technology to increase the proportion of minority samples, and improve the classification accuracy of minority classes and the overall performance of classifiers.

At present, there are some methods such as changing the distribution of data and improving the existing classification algorithms to address the above problems from different perspectives, which achieve good results [2,15]. The most common strategies for changing the distribution of data to achieve data balance are over-sampling and random under-sampling. Random oversampling is to copy a small number of samples to balance the sample size of the dataset. Random under-sampling uses a certain strategy to select a subset of the majority samples for the same purpose. Literature [8] proposed an imbalanced data classification method based on RSBoost algorithm. This method uses the SMOTE algorithm to minority class sample, and then randomly under-samples the original dataset to improve the imbalance of the original dataset and minority class of classification accuracy. Literature [7] proposed a new imbalanced data learning algorithm PCBoost, which uses the data synthesis method to add the synthesized minority samples and balance the training samples. This improves the existing classification algorithms such as cost-sensitive learning, ensemble learning, single-class learning and so on; selective ensemble learning has been widely studied.

* Corresponding author.
E-mail address: tjoman@126.com

The above approach can solve the problem of imbalanced data to some extent in certain environment [4,6,10]. Nevertheless, in order to obtain balanced data, sampling by immutable proportion would undermine the distribution characteristics of the original data, and sampling by lessening scale in the majority may remove the potentially useful information for classification, which would lead to low efficiency, low classification accuracy and other troubles. In addition, with the increase of data size and the computational cost brought by the algorithm for the preprocessing imbalanced data, the classification efficiency has become a bottleneck.

The emergence of cloud platform and parallelization technology has provided us with a convenience to bridge the classification efficiency gap and achieved effective results [1,11,12,13]. Literature [15] proposed weight-distributed machine learning based on MapReduce framework for imbalanced data. The experimental results show that this method has better scalability and higher processing efficiency for imbalanced big data. However, MapReduce is not suitable for iterative algorithms due to high network and I/O overhead in writing intermediate output to disk and reading data back from disk. As compensation for this deficiency, Apache Spark is a memory-based, fast-processing big data framework that solves the problems of iterative computation and interactive data mining. Literature [16] proposed a short text feature extension and classification method based on association rule mining on the Spark platform. The experimental results show that the average efficiency improvement reaches 15% compared with the traditional classification method. Literature [3] proposed a parallel stochastic forest algorithm for Spark based big data processing, and a large number of experimental results show its advantage in classification accuracy and efficiency. In Spark platform, those methods mentioned above are only effective for balanced data. Scarce literature has been witnessed to make full use of the computational advantages of Spark framework to solve the problem of unbalanced data classification efficiency.

To overcome the bottleneck and bridge this gap, we introduce a Spark-based Ensemble Learning for imbalanced data classification (SELidc in short) method. SELidc firstly converts the original imbalanced dataset into resilient distributed datasets. Next, in the sampling process, it samples by comprehensive weight, which is obtained in accordance with the weight of each class in majority class and the number of minority class samples. After that, it trains several classifiers with random forest in Spark environment by the correlation feature selection means.

2. The SELidc Method

The Random forest [2] (RF) is an ensemble learning algorithm, which builds a flock of independent and distinct decision trees based on the idea of randomization. It applies the bootstrap resampling technique. Firstly, extract randomly *k* new self-sample sets from the original training sample set, and construct *k* tree categories (usually using a decision tree). The samples that have not been drawn up are composed of *k* out-bag-data at every turn. Next, the generated trees compose a random forest. After that, the random forest classifier is used to distinguish and classify the new data, and the classification results are determined according to the voting results of each classifier. The random forest is good at ensemble classification.

Nevertheless, the random forest algorithm also has its shortcomings: poor classification accuracy for imbalanced datasets and the random in feature selection. To overcome those shortcomings, SELidc mainly improves the random forest algorithm from two aspects. On one hand, a balancing-data method based on comprehensive weight is proposed to balance samples; On the other hand, it changes the traditional way of selecting features randomly. Both are implemented in a distributed environment. In SELidc method, not all samples are used as input samples of each tree, which not only ensures the difference between training base classifiers but also avoids overfitting.

2.1. Data-balancing tactics

In order to achieve data balance and keep the distribution of original data and to mine more valuable data in majority samples, SELidc adopts a data-balancing tactics based on comprehensive weight. It firstly calculates the comprehensive weight according to the weight of each class in majority class and the number of minority class samples. Next, it samples the majority samples in accordance with the ratio of the comprehensive weight. The times of sampling is determined by the sample number of minority and majority class. After that, it combines them with the minority samples so that the proportion of minority class in all training samples becomes bigger. Finally, the combined subset of training samples is used as input sample data of random forest, and the basic definitions of the above contents are as follows.

Definition 1. Let N arbitrary distinct samples with M features, samples and class are denoted with collection X , Y respectively, where $X=\{a_1,a_2,...,a_N\}$ and $Y=\{y_1,y_2,...,y_N\}$, then training sample are mathematically modeled as matrix Z as shown in Equation (1).

$$Z=[X,Y]=\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1M} & y_1 \\ x_{21} & x_{22} & \cdots & x_{2M} & y_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{N1} & x_{N2} & \cdots & x_{NM} & y_N \end{bmatrix}_{N\times(M+1)} \tag{1}$$

Specifically, where $a_i=\{x_{i1},x_{i2},...,x_{ij},...,x_{iM}\}$ represents the n features of the *i*th sample, it x_{ij} represents the *j*th feature of sample a_i.

Definition 2. Suppose a certain class of sample is the least, then the samples as the minority class sample, named *M*inority$(Minority\subset X)$, the number of *M*inority named p, the last samples as the majority class sample, named $Majority(Majority\subset X)$, the number of *Majo*rity named $m-p$, so the sampling times is $\kappa=(m-p)/p$.

Definition 3. For the majority class distribution is $T_1,T_2,...,T_r$, the number of sample for each class is called $Count(T_1),Count(T_2),...,Count(T_r)$, such that sampling weight of each class as shown in Equation (2).

$$Ration(T_i)=1-\frac{Count(T_i)}{\sum_{i=1}^{r}Count(T_i)} \tag{2}$$

Note that, the smaller the sample size, the greater the sampling weight. The number of minority class samples is called *p*, the comprehensive weight $Weight(T_i)$ of the majority class T_i as shown in Equation (3).

$$Weight(T_i)=p\times\frac{Ration(T_i)}{\sum_{i=1}^{r}Ration(T_i)} \tag{3}$$

2.2. Feature selection

As traditional means of Random Forest, feature subsets are selected in a random way so the accuracy and the error rate usually become worse in the face of noise and redundant datasets. These noise and redundant features do not contribute to the accuracy of classification, and their importance is low. Therefore, based on the literature [17], SELidc adopts a new correlation-based feature selection means to remove the noise and redundant features. Feature selection from the beginning of the empty set, select the features that have the ability to distinguish between classes in turn, and select the best *k* features. Because it is not in the math model and it can't be directly obtained from the original data, make it as a hyper-parameter, through cross validation. We set *k* as the number of features corresponding to the highest performance of the classification model.

Definition 4. Set training sample scale for *N*, the number of class for *l*, which is $\{(a_k,y_k)|y_k\in\{1,2,...l\},k=1,2,...N\}$.

Where the number of samples in class *j* is n_j, then the formula for distinguishing degree F_i between classes of the feature *i* is as shown in Equation (4).

$$F_i=\frac{\sum_{j=1}^{l}(x^{(j)}-x)^2}{\sum_{j=1}^{l}\frac{1}{n_j-1}\sum_{k=1}^{n_j}(x_{k,i}^{j}-x_i^{(j)})^2} \tag{4}$$

Note that, where x and x^j represent the mean value of feature x on the whole dataset and the *j*th dataset respectively; $x_{k,i}^{(j)}$ is the feature value of the feature of the *k*th sample in *j*th class. The Equation (4) shows the ratio of the distance

between class of feature x and the variance within the class, the larger the value, the stronger the classification recognition of features x.

The process of the SELidc is constructed as shown in Figure 1, where k is the number of decision trees in the random forest, N is the number of samples in the training dataset corresponding to each decision tree, M is the number of features in the sample, X is the number of the training classifier, u is the number of features which fit $u < m$.

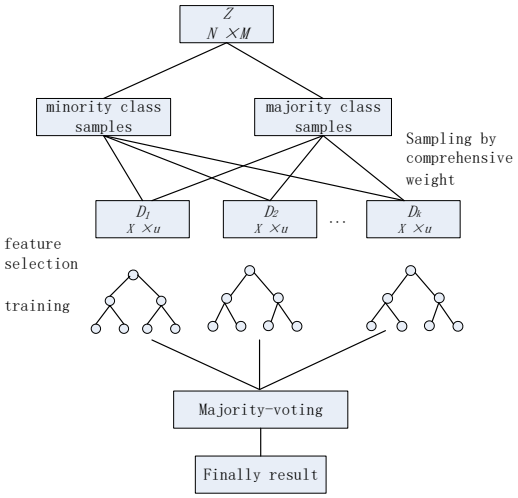

Figure 1. Process of the SELidc

3. Algorithm implementation of SELidc in Spark

3.1. About Spark

Apache Spark, a distributed computing framework based on memory computing which developed originally at UC Berkeley's AMP Lab, is a memory distributed computing framework. Different from Hadoop storing output in HDFS, its intermediate output and the result stores directly in memory. Spark proposed the concept of Resilient Distributed Dataset (RDD), which is a set of read-only objects distributed in a cluster and partitioned into multiple nodes in a cluster. In order to re-use conveniently datasets for the next calculation, it caches datasets in memory, provides faster access to data, reduces unnecessary duplication of I/O. Spark not only supports dataset-based applications, but also has features such as fault tolerance, local compute scheduling, and scalability. Therefore, Spark is better suited for areas such as big data mining.

3.2. Algorithm design and implementation

The idea of SELidc consists of the following three points: raising the proportion of minority class; keeping the distribution of original data and mining more valuable information in majority class; reducing the overhead of I/O. Its main stages as follow: 1) converts the original imbalanced dataset into resilient distributed datasets; 2) samples in accordance with the comprehensive weight which obtained by the Equation (2) and (3); 3) selects features according to Equation (4) and trains several classifiers with random forest in Spark environment; finally, obtains the final classification result by weighted voting.

Phase 1: Converting the original imbalanced dataset into resilient distributed datasets. The process of converting from original dataset to *RDD* is shown in Figure 2. Firstly, read the training set data from HDFS distributed system by using *sparkContext. TextFile* function and convert it into the form of *RDD*. Each data record is mapped into tuple form (key, value) by *map ()* function. The key for the sample class is represented a record. Then, calculate the sample size for each class by calling the collect () function and call the sortBy () function to sort in ascending order, and finally get and will be used for the sampling phase of the next stage. The algorithm based on SELidc as shown in Algorithm 1.

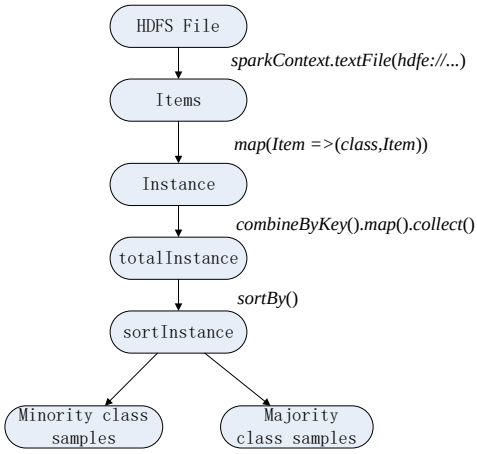

Figure 2. Conversion Lineage graph in phase 1

Conversion algorithm in Phase1

//Input： *D*, raw data
//Output： RDD_{min}, RDD_{maj}
1. for each sample x in *D*
2. *instance*←x.map(class,x)
3. *totalInstance*←instance.combineByKey()
.map(num+1).colletct()
4. *sortInstance* ← totalInstance.sortBy(num,false)
5. RDD_{min}←*sortInstance* (0)
6. RDD_{maj} ←*sortInstance* (1)

Algorithm1. Conversion algorithm based on SELidc

Phase 2: Sampling based on comprehensive weight. In order to achieve data balance and keep the distribution of original data and mine more valuable information in majority class samples, SELidc samples in accordance with the comprehensive weight which obtained by Formula (3), and its specific process as shown in Algorithm 2.

Sampling algorithm in Phase2

//Input： RDD_{min}， RDD_{maj}
C， numbers of classes
//Output： *RDDi*, a balanced train set
1. $C_1,C_2,...,C_n$← n classes
2. sum ← numbers of RDD_{maj} sample
3. p ← numbers of RDD_{min} sample
4. M ← numbers of train subsets
5. for each class C_i
6. if key ==C_i then
7. *num*++
8. *Ration*(C_i)， *Weight*(C_i)← according the definitions calculate it
9. RDD_i ← select samples into RDD_i
10. RDD_i ← $RDD_i \cup RDD_{min}(i \in [1,M])$

Algorithm 2. Sampling algorithm based on SELidc

Phase 3: Training classifier and weighted voting decision. The main purpose of this stage is to train sub-classifier and obtain classification model by using the training sets and feature subsets processed in the above two stages. The traditional random forest algorithm adopts the simple voting principle and gives the same weight to each decision tree; it neglects the difference between the strong classifier and the weak classifier and cuts down the overall classification performance. In this paper, SELidc adopts the weighted voting method to generate the final result. Its main idea is using the unsampled data as out-of-pocket (OOB) data to calculate the accuracy of each decision tree classifier, and use the accuracy as the voting weight of the corresponding decision tree. Its flow chart is presented in Figure 3.

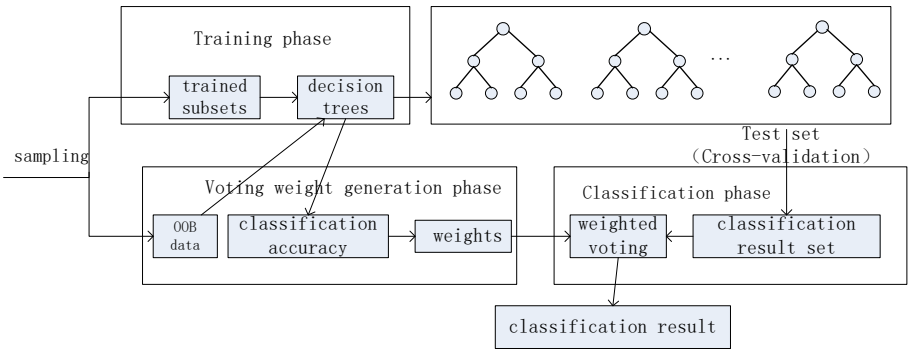

Figure 3. Flow chart of Phase3

4. Experimental results and analysis

The experimental environment based on Spark cluster, including 1 master and 7 slave nodes, all nodes of the configuration is as follows: the operating system is Centos 7, CPU Intel Core i7-6700 CPU @2.60Ghz 2.59Ghz, 8GB memory, using Scala 2.10.4 development, cluster environment based on Hadoop2.6.0 Spark1.6.0.

4.1. Experimental setup

In order to evaluate the effectiveness of SELidc in the classification of unbalanced datasets, six datasets with imbalanced were chosen for experiments. The dataset is from the dataset of the UCI machine learning database. Among them, the UCI dataset is shown in Table 1, the sample number of UCI dataset is #Ex, the #F is the characteristic number, the # Minority is the minority sample, the #R is the proportion of minority sample in dataset. In order to verify the effectiveness of the proposed method in the spark platform, the KDD Cup 99 dataset [11] is used, its whole training data has 500000 records and the URL Reputation dataset, with 2396130 training data and 2.05GB data capacity. In order to evaluate the performance of the proposed algorithm, we compare the decision tree C4.5 algorithm with the random forest RF algorithm. The comparison algorithm runs under the Spark platform to ensure the consistency of experimental operating environment. In this paper, the training base classifier using C4.5 algorithm, which iterate 20 times; C4.5 decision tree algorithm directly to the imbalanced datasets classification, the experiment selected the same number of features and base classifier algorithm using C4.5 algorithm.

Table 1. Summary description of the used datasets

Datasets	#Ex	#F	#C	#R
Balance-scale	625	4	3	7.84
Satimage	6435	36	6	9.7
Segment	2310	19	7	14.29
Sick	3772	29	2	6.12
Vehicle	846	18	4	25.06
Yeast	1484	8	10	3.56

4.2. Evaluation indicators

In this paper, the recall rate, accuracy and F-measure are used as indexes to evaluate the performance of the classifier. These three indicators are widely used in data mining and other fields. The traditional dichotomous evaluation indicators are based on the confusion matrix, in which the positive class and the negative class represent the minority class and the majority class respectively in the imbalanced datasets. TP and TN denote the number of positive and negative samples correctly classified respectively; FP denotes the number of samples misclassified as positive; FN denotes the number of samples misclassified as negative and shown in Table 2.

Table 2. Confusion matrix for a two-class problem

	Positive	Negative
Positive	TP	FN
Negative	FP	TN

The popular metric that is used in an imbalanced class scenario is the $F_{measure}$, which is defined in Equation (5).

$$F_{measure} = \frac{(1+\beta^2)\times Presicion \times Recall}{\beta^2 \times (Presicion + Recall)} \tag{5}$$

Where $Recall = TP/(TP+FN)$ and $Precision = TP/(TP+FP)$, β which used to adjust the relative importance of Recall and Precision. In this work, giving it a value of 1.

4.3. Experimental results

Experiment 1. Benefits of sampling by comprehensive weight.

In the imbalanced datasets, the number of majority samples is far more than that of the minority ones. The traditional classification algorithms are insensitive to minority class. For example, even dividing all samples into majority classes, it still achieved high classification accuracy. However, the minority class is always unrecognized by most of the traditional classification algorithms. So, it is very important to increase the proportion of minority class in training samples.

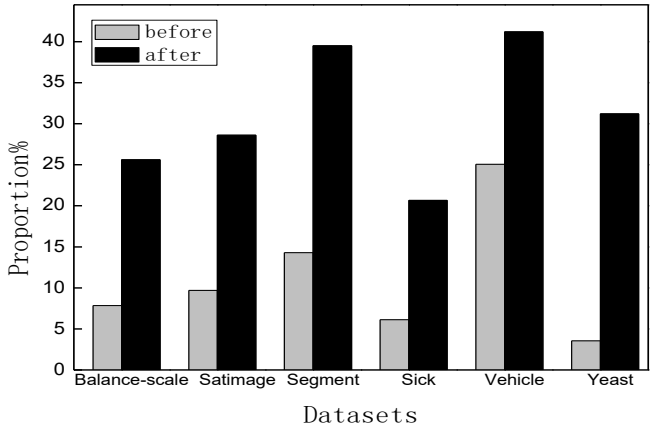

Figure 4. Effect on improving proportion of minority class

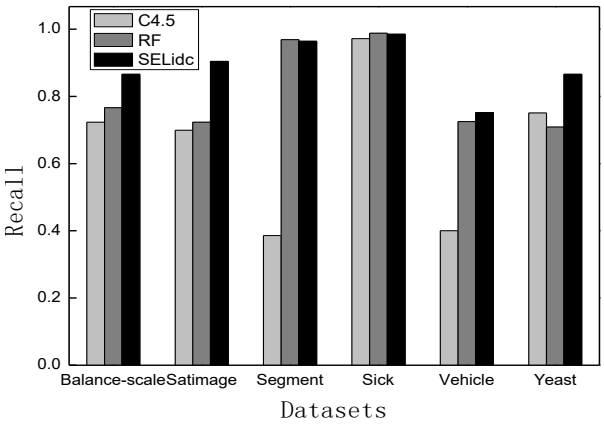

Figure 5. Recall of the different algorithms

It can be seen from Figure 4 that the proportion of the minority class in each training sample is obviously improved after the imbalanced datasets processed by SELidc.

Experiment 2 Results on different datasets and algorithms.

In the process of classification, C4.5, random forest algorithm and the SELidc algorithm are respectively designed for three groups of experiments, each of which is tested separately for the above dataset. According to the Equation (5), the experimental results have been shown in Table 3 and its corresponding figures are presented as follows.

Table 3. Results on different datasets and algorithms

Datasets	methods	$Recall$	$Precision$	$F_{measure}$
Balance-scale	C4.5	0.723	0.667	0.694
	RF	0.766	0.732	0.749
	SELidc	0.866	0.825	0.842
Satimage	C4.5	0.699	0.691	0.694
	RF	0.723	0.711	0.716
	SELidc	0.904	0.833	0.867
Segment	C4.5	0.386	0.286	0.386
	RF	0.969	0.969	0.969
	SELidc	0.964	0.964	0.964
Sick	C4.5	0.972	0.972	0.972
	RF	0.988	0.988	0.889
	SELidc	0.985	0.985	0.985
Vehicle	C4.5	0.4	0.384	0.413
	RF	0.725	0.722	0.722
	SELidc	0.752	0.752	0.744
Yeast	C4.5	0.751	0.751	0.745
	RF	0.709	0.709	0.71
	SELidc	0.866	0.826	0.842

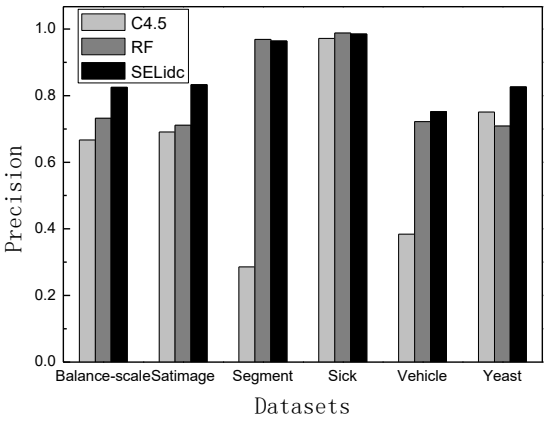

Figure 6. Precision of the different algorithms

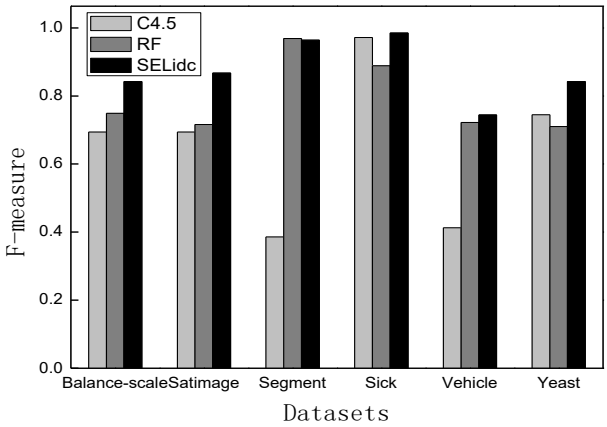

Figure 7. Results of the different algorithms

According to Figure 5 and Figure 6, it can be seen that the accuracy of C4.5 algorithm is low on Segment and Vehicle datasets while RF algorithm has higher recall and precision on Segment and Sick datasets. Compared with the RF and C4.5 algorithms, the accuracy of SELidc algorithm in the Balance-scale, Satimage and Yeast datasets respectively has significant advantages. SELidc outperforms the C4.5 algorithm on these six datasets, and is roughly the same with the RF algorithm on the Segment and Sick datasets. Figure 7 is a comparison of three different methods in evaluating the performance of unbalanced data classification.

Only when the recall rate and accuracy are become larger does F-measure becomes correspondingly larger. Therefore, it is reasonable to evaluate the classifier's classification performance for minority class. Compared with the other two algorithms, the classification performance of SELidc is greatly improved. For example, the F-measure value of SELidc in Balance-scale, Satimage and Yeast datasets is 10% higher than that of the other two algorithms.

Experiment 3. Effect of decision tree numbers on classification accuracy

In order to verify the effect of the number of decision trees on classification accuracy, we set different decision tree numbers {5, 10, 15, 20, 25, 30}. The average classification accuracy of RF, C4.5 and SELidc are shown in Figure 8.

As seen from Figure 8, the number of decision trees plays an important role in improving the classification accuracy. Starting from 5 decision trees, SELidc always gets the better performance than the other two algorithms. With the numbers of decision tree creasing, the growth rate of all these three algorithms has slowed down, which means that the number of decision trees is optimal. When the number of decision tree grows to 20, the average classification accuracy of the SELidc algorithm reaches its peak.

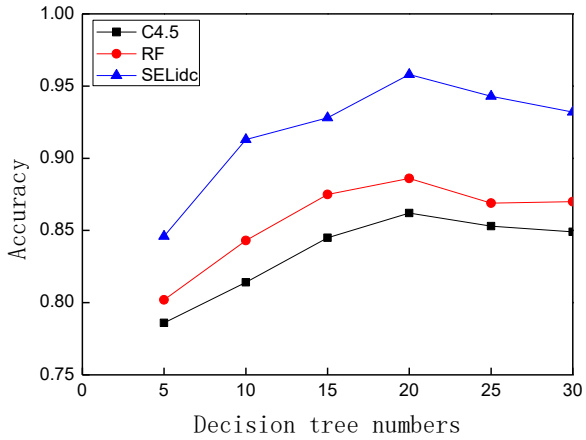

Figure 8. Effect of decision tree numbers on accuracy

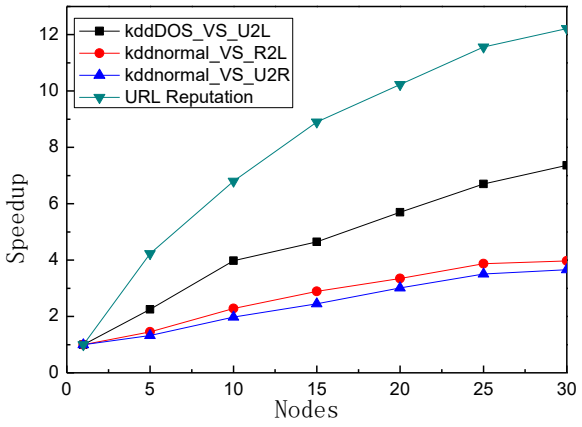

Figure 9. Speedup on different scale datasets

Experiment 4. Execution efficiency on different scale datasets

In order to verify the efficiency, SELidc is compared with the traditional single on-machine classification algorithm and tested on KDD Cup 1999 experimental dataset and UCI datasets URL Reputation dataset; the speedup of the algorithm in this paper varies with the number of nodes in the cluster. The number of nodes is {1,5,10,15,20,15,30}. Algorithm speedup, which is the algorithm in a stand-alone environment and distributed environment running time ratio, is used to evaluate the effect of the parallel program. Figure 9 is the SELidc algorithm speedup experiment results.

From the above experimental results showed in Figure 9, it can be seen that the speedup of the acceleration ratio increases linearly and steadily on this dataset while the computing nodes of the spark cluster increase, especially, it grows obviously on datasets kddnormal_VS_U2R and kddnormal_VS_R2L. This is mainly due to the dataset capacity is too small, and the algorithm's distributed processing capabilities are not fully utilized. On the other hand, in the case of the same node, the curve grows more obviously when the dataset becomes larger, and the speed performance of the algorithm gets better. Thus, it can be seen that the classifier training under the Spark environment has better parallelization performance. So, when training massive data, it is possible to reduce the time required to train the model greatly by increasing the number of computing nodes in the spark cluster.

5. Conclusions and future work

In order to maintain the distribution of original data, make better use of potentially useful classification information, and ensure the differences among the classifiers in the ensemble learning, we introduce a Spark-based Ensemble Learning for imbalanced data classification (SELidc) method to improve performance from two aspects. One aspect is to preprocess in order to balance the imbalanced datasets by sampling with comprehensive weight. The other one is to reduce fitting for the big and imbalanced data by building distributed ensemble learning algorithm. Experimental results on various datasets collected from UCI and other datasets demonstrate that SELidc achieves better performance than other related methods, and makes full use of the efficient computing power of Spark distributed platform.

Acknowledgements

The paper is supported by Grants from the National Natural Science Foundation of China (No. 51467007, 61562054, 61462050).

References

1. A. Andrzejak, F. Langner, S. Zabala, "Interpretable Models from Distributed Data via Merging of Decision Trees," in IEEE International Conference on Computational Intelligence and Data Mining, pp.1-9,2013.
2. L. Breiman, “Random forests,” Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.
3. J. Chen, K. Li, Z. Tang, K. Bilal, S. Yu, C. Weng and C. Li, "A Parallel Random Forest Algorithm for Big Data in a Spark Cloud Computing Environment," in IEEE Transactions on Parallel & Distributed Systems, vol. 28, no. 4, pp. 919-933.2017.
4. X. Hu, J. Wen, Y. Zhong, "Imbalanced Data Ensemble Classification using Dynamic Balance Sampling," presented at the CAAI transactions on intelligent systems, 2016.
5. H. Kim, P. Howland, H. Park, "Dimension Reduction in Text Classification with Support Vector Machine," Journal of Machine Learning Research, vol.6, no. 1, pp. 37-53,2005.
6. Q. Q. Li, X. Y. Liu, "EasyEnsemble.M for Multiclass Imbalance Problem," Pattern Recognition and Artificial Intelligence, vol27, no.2, pp. 187-192, 2014.
7. X. F. Li, J. Li, Y. F. Dong, “A New Learning Algorithm for Imbalanced Data—PCBoost," Chinese journal of computers, vol. 35, no. 2, pp. 2202-2209, 2012.
8. K. W. Li, L. Yang, W. Y. Liu," Classification Method of Imbalanced Data Based on RSBoost," Computer Science, vol. 42, no. 9, pp. 249-252, 2015.
9. S. Liu, J. Zhou, B. Li, "Entity Relation Extraction Method Based on Multi-SVM-KNN Classifier," Journal of Data Acquisition and Processing, vol. 30, no. 1, pp. 202-210, 2015.
10. B. Ma, Y. Zhou, J. J. He, "Variational Gaussian Process Classification Algorithm for Large-scale Class-imbalanced Data," Journal of Dalian University of Technology, vol. 56,no. 3, pp. 279-284, 2016.
11. J. Qin, X. Qian, W. Wang, "An Algorithm for Unbalanced Big Data Using Paralleled Random Forest," Microelectronics & Computer, vol. 34, no. 4, pp. 22–27, 2017.
12. S. del. Rio, V. Lopez, J. M. Benitez, and F. Herrera, "On the Use of MapReduce for Imbalanced Big Data Using Random Forest," Inform. Sci., vol. 285, pp. 112–137, 2014.
13. P. K. Ray, S. R. Mohanty, N. Kishor, and J. P. S. Catalao, "Optimal Feature and Decision Tree-based Classification of Power Quality Disturbances in Distributed Generation Systems," in IEEE Trans. Sustain. Energy, vol. 5, no. 1, pp. 200–208, 2014.
14. X. Tao, S. Hao, D. Zhang, P. Xu, “Overview of Classification Algorithms for Unbalanced Data," Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition)., vol. 25, no. 1, pp. 102-110, 2015.
15. Z. Wang, J. Xin, S. Tian, G. Yu, "Distributed Weighted Extreme Learning Machine for Big Imbalanced Data Learning," Proceedings of ELM-2015 Volume 1. Springer International Publishing, 2016.
16. W. Wang, K. Zhao, C. Li, "Feature Extension and Category Research for Short Text Based on Spark Platform," Journal of Frontiers of Computer Science and Technology, vol. 11, no. 5, pp. 732–741. 2017.
17. J. Xie, W. Xie, "Several Feature Selection Algorithms Based on the Discernibility of a Feature Subset and Support Vector Machines," Chinese Journal of Computers, vol. 37, no. 8, pp. 1704–1718, 2014.
18. Q. Zhou，M. Guo, Y. Liu, "A Classification Method for Class—Imbalanced Data and Its Application on Bioinformatics," Journal of Computer Research and Development, vol. 47, no. 8, pp. 1407-1414, 2010.

Jiaman Ding is an Associate Professor in the Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, China. He now is also a Ph.D. candidate in Kunming University of Science and Technology. His current research interests include data mining, cloud computing, machine learning.

Sichen Wang is an MPhil student in the Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, China. Her current research interests include machine learning, data mining.

Lianyin Jia is an Associate Professor in the Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, China. He received his Ph.D. degree in Computer Science from South China University of Technology, Guangzhou, China in 2013. His current research interests include database, data mining, information retrieval and parallel computing.

Jinguo You is an Associate Professor in the Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, China. He received his Ph.D. degree in Computer Science from South China University of Technology, Guangzhou, China in 2009. His current research interests include data warehouse and data mining.

Ying Jiang is a Professor in the Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, China. She received her Ph.D. degree in Computer Software and Theory from Peking University, Beijing, China in 2005. Her current research interests include data mining and software engineering.