Association rule mining algorithm based on Spark for pesticide transaction data analyses

Xiaoning Bai[1,2], Jingdun Jia[1,3*], Qiwen Wei[4], Shuaiqi Huang[1], Weicheng Du[5], Wanlin Gao[1*]

(1. *College of Information and Electrical Engineering, China Agricultural University, Beijing 100083, China*;
2. *Institute for the Control of Agrochemicals, Ministry of Agriculture and Rural Affairs, Beijing 100125, China*;
3. *Ministry of Science and Technology Torch Center, Beijing 100045, China*;
4. *National Agricultural Technology Promotion Center, Beijing 100125, China*;
5. *Information Center, Ministry of Agriculture and Rural Affairs, Beijing 100125, China*)

Abstract: With the development of smart agriculture, the accumulation of data in the field of pesticide regulation has a certain scale. The pesticide transaction data collected by the Pesticide National Data Center alone produces more than 10 million records daily. However, due to the backward technical means, the existing pesticide supervision data lack deep mining and usage. The Apriori algorithm is one of the classic algorithms in association rule mining, but it needs to traverse the transaction database multiple times, which will cause an extra IO burden. Spark is an emerging big data parallel computing framework with advantages such as memory computing and flexible distributed data sets. Compared with the Hadoop MapReduce computing framework, IO performance was greatly improved. Therefore, this paper proposed an improved Apriori algorithm based on Spark framework, ICAMA. The MapReduce process was used to support the candidate set and then to generate the candidate set. After experimental comparison, when the data volume exceeds 250 Mb, the performance of Spark-based Apriori algorithm was 20% higher than that of the traditional Hadoop-based Apriori algorithm, and with the increase of data volume, the performance improvement was more obvious.

Keywords: Spark, association rule mining, ICAMA algorithm, big data, pesticide regulation, MapReduce
DOI: 10.25165/j.ijabe.20191205.4881

Citation: Bai X N, Jia J D, Wei Q W, Huang S Q, Du W C, Gao W L. Association rule mining algorithm based on Spark for pesticide transaction data analyses. Int J Agric & Biol Eng, 2019; 12(5): 162–166.

1 Introduction

Smart agriculture is a modern agricultural mode supported by Internet of Things technology and data science. It is the outcome that combines information technology and agriculture compared with precision agriculture, smart agriculture focuses on how to make the most efficient use of various agricultural resources and minimize agricultural energy consumption, including smart production, smart circulation, smart sale, smart community and smart management. As of 2018, Chinese digital economy[1] ranks second in the world and Chinese agriculture has entered the age of digitalization. China has gradually achieved that information perception, quantitative decision-making, intelligent control and personalized service in the whole process of agricultural production. In this context, agricultural operators have a huge demand for agricultural-related information services in order to achieve the accurate investment of agricultural inputs[2]. Pesticide is an important agricultural input. China's pesticide application ranks first in the world, far higher than the average level of pesticide application in the world, posing a serious threat to wildlife, soil and water resources[3]. At present, the agricultural supervision data of China has accumulated a certain scale, and the pesticide transaction data collected by the pesticide national data center only produces more than 10 million records daily. In order to solve the problem of pesticide abuse, it is urgent to mine the hidden relationship in the pesticide circulation data. In turn, so as to provide data support for the supervision and management and healthy development of the pesticide industry.

Spark is a parallel computing framework based on In-memory cluster computing, which has a one hundred times better performance than the popular Hadoop MapReduce algorithm. It ensures the real-time performance of data processing in the big data environment with high fault-tolerance and high scalability. Thus, this framework is commonly used for analyzing mass data because of its excellent performance[4]. Spark’s memory computing is based on a new distributed memory abstract resilient distributed dataset (RDD). For RDD, Spark has many built-in operations that can convert one RDD to another. Memory calculations are made up of this series of RDD operations. In particular, the RDD persistence operation can cache the RDD in the memory of the working node[5], so that when the subsequent operations reuse the data, they can be directly read from the memory. This is another factor affecting the computing speed of Spark. In addition, Spark's fault-tolerant approach is also very different from Hadoop, which is fault-tolerant through multiple copies of data. Spark does not need to back up data. It records a series of operations performed on the RDD and constructs a directed acyclic graph

Received date: 2018-12-23 **Accepted date:** 2019-05-22
Biographies: Xiaoning Bai, PhD candidate, research interests: agricultural informatization. Email: bxn@agri.gov.cn; **Qiwen Wei,** PhD, Professor, research interests: agricultural plant protection. Email: weiqiwen@agri.gov.cn; **Shuaiqi Huang,** Master, research interests: data mining. Email: caucieehsq@163.com; **Weicheng Du,** MS candidate, research interests: agricultural informatization. Email: 1277170016@qq.com;
***Corresponding author: Jingdun Jia,** PhD, Professor, research interests: agricultural informatization. No.2 Sanlihe, Xicheng District, Beijing 100045, China. Tel: +86-18601058700, Email: jiajd@most.cn; **Wanlin Gao**, PhD, Professor, research interests: intelligent agriculture. College of Information and Electrical Engineering, China Agricultural University, Beijing 100083, China. Tel: +86-10-62736755, Email: gaowlin@edu.cn.

(DAG). If the data is in error or lost, it is recalculated according to the DAG.

Spark was originally developed as a cluster-computing framework by University of California, Berkeley in 2009, then became open-source next year. There was a lack of data mining framework at that time, and most of these frameworks available were insufficient in optimization. Apriori algorithm, which was proposed by Agrawal in 1993, is mainly used for association analysis. The algorithm is able to obtain frequent itemsets by generating candidate itemsets and testing downward closure lemma[6].

Some modifications were proposed to develop Apriori algorithm. Lin et al.[7] proposed an improved Apriori algorithm based on array vector, which reduces the number of connection and unnecessary traversing, improves the utilization efficiency of memory. Similarly, the improved Apriori algorithm based on vector matrix was proposed by Cao et al.[8] Zhao et al.[9] used orthogonal linked list to improve the storage process of Apriori algorithm. This algorithm simplifies the Scala process and the pruning process, thus simplifying the generation process of frequent itemsets and improving the time efficiency of Apriori algorithm.

TF-IDF is another Association Rules Mining algorithm. It is used for feature extraction in text based on Vector Space Model by calculation the weight of each feature item for the text, to extract the key words and core content in the article. In addition, TF-IDF is able to be used of dimension reduction of features for text preprocessing[10]. Therefore, based on the existing research, this paper proposed an optimal Apriori algorithm and implemented parallelization based on Spark. From the experiments on data sets of millions of orders and analysis of the algorithms idea and performance, we find that there are three deficiencies in the Apriori algorithm. (1) The method of filtering out non-frequent itemsets in processing of generating the L_{k+1} needs further improvement. (2) There is simplified margin for excessive connections in itemsets during the processing of L_k connection. (3) Apriori algorithm will access many redundant data items and transactions when traversing the database. In view of the weakness of Apriori algorithm mentioned above, the corresponding improvement methods, and more efficient algorithm ICAMA were proposed in this paper. The ICAMA algorithm uses the idea of MapReduce to improve the two stages of Apriori. The first stage: the data structure is changed while reading the data set to be processed from the HDFS, and the first frequent item set is filtered, and the finally obtained data set is stored in the RDD form in the memory of each node of the cluster. The second stage: frequent k itemsets are directly generated based on frequent k–1 item sets. Repeat this process until no more frequent itemsets are generated[11]

They all include the processing that data transformation and statistics. A comparison experiment between ICAMA algorithm and MapReduce based Apriori algorithm shows a result of 20% performance improvement for B-Apriori. And the high performance is maintained even dealing with a million-level dataset. In addition, this paper implements ICAMA algorithm based on Spark framework, which fills the gap that there is no algorithm for Association Rules Mining in Spark's scalable machine learning library (MLlib) [12].

2 Design and implementation

2.1 Introduction of Spark framework

Spark is a parallel computing framework originally developed by the Berkeley AMP laboratory, which is based on In-memory cluster computing. This framework has the advantage of in-memory computing based on Resilient Distributed Dataset (RDD), so it is faster than Hadoop MapReduce computing framework. In-memory computing is the key to the high efficiency of Spark framework, which refers to loading useful data onto the database into the memory of the computing node when Spark is working. RDD is the implementation of in-memory computing. Persist operation and fault tolerance are two significant characters of RDD[13]. The effect of persist operation is to cache the RDD to memory of the computing node. So, persist operation provides more inefficient procession in reuse data. Unlike Hadoop, Spark builds DAG by recording historical operations on RDD to improve fault tolerance instead of data backup. When data is wrong or lost, Spark gets correct RDD by original RDD and tracing the DAG.

Spark improves performance by 100 times compares to traditional Hadoop MapReduce method. The architecture of Spark shown as figure 1can be divided into four modules: Spark SQL-RDD (for unit of data execution), MLlib (for Machine leaning), Graphx (for graphs computation) and Spark Streaming (for real-time processing). Meanwhile, Spark is highly efficient because it’s able to store intermediate results of iterations in memory rather than in hard disk. The modules of Spark will be described in following[14].

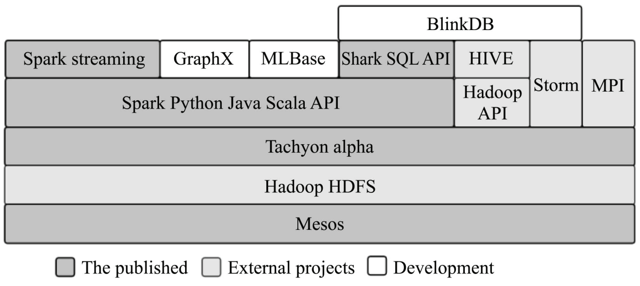

Figure 1 Architecture of Spark

2.2 Algorithmic details of Apriori

It only requires traversing the data set twice for Spark to implement the matrix-based Apriori algorithm. Combining with technological architecture, Spark improves the efficiency of Association Rules Mining by using global and local support-based pruning. Its transaction data sets and frequent itemsets are stored in HDFS file system based on Hadoop. In order to save memory space and reduce traversing times, the matrix stores Boolean values, and each row as a transaction, each column as a differential item. The support counts of itemsets can be got by doing “and’ operations between corresponding matrices[15].

Apriori is an important algorithm for Association Rules Mining. It can be divided into two steps: the first step is to find all the frequent itemsets and the second step is to generate association rules based on frequent itemsets. When the number of sets is greater than 0, a list of candidate itemsets consisting of k items is generated, and then to keep frequent itemsets and generate a list of candidate itemsets consisting of k+1 items[16].

2.3 Implementation of distributed Apriori based on Spark

This paper implements a distributed Apriori algorithm using Scala programming language, which mainly combines Spark framework and RDD operator. The implementation of the algorithm is divided into the following two parts.

The first part is to generate frequent itemsets L_1, which is shown in Figure 2. Including:

1) Use *flatMap* to let transaction set T be distributed to parallel computing system in the form of RDD<String and Number>.

2) Accumulate number of items with *reduceByKey*.

3) Use *filter* to filter down the item set less than the support.

The second part,is to get L_K from L_{K+1}. Including:

1) L_K Self-join to C_{K+1}.

2) Traverse the database, compare CK by the method in first part.

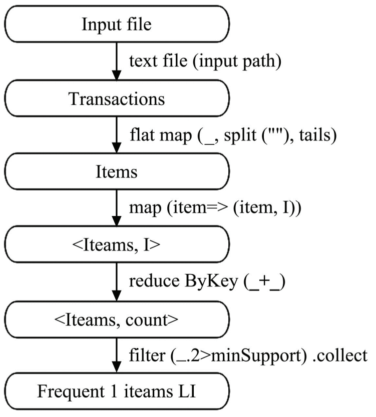

Figure 2 Flowchart of distributed Apriori

3 Improvement of ICAMA algorithm

3.1 The idea of improving the algorithm

In the first phase of the classic Apriori Spark-based implementation YAFIM, the data set to be processed directly into the HDFS[17] in the first stage is stored in the RDD form in the memory of each node of the cluster, and then each map task reads in and processes several rows. Each item contained in these lines is transmitted with a value of 1, and the reducer sums and filters all the value values of each item to obtain an item whose number of occurrences is not less than the minimum support. Since the data set is read directly from HDFS, its organization in memory remains the same: each row represents a transaction T, and T consists of the TID[18] and all the items contained in the transaction T. Therefore, it is necessary to calculate the number of occurrences of an item set, and only the data set can be traversed as a whole to count the results. This process is repeated for each iteration, which is a considerable time consumption.

In order to solve the above problems, ICAMA adopts a data structure conversion method to read data sets from HDFS and realize data structure conversion[19]. Each map task reads in and processes several rows. The processing method is different from YAFIM[20], but is included in the transaction. All items emit the key-value pairs of the item and the corresponding transaction number, and the reducer combines each corresponding transaction number. The transaction number is then counted and filtered to obtain a converted data set F containing only a frequent set of 1 items. The structure conversion process of the data set[21].

A_{xy} in F represents whether Ix is included in TID_y. If it is included, a_{xy} is 1, otherwise it is 0. At this time, if you need to calculate the number of occurrences of a k item set in the entire data set, you only need to find the value corresponding to the k items in data set F. The result of the operation is all the transaction numbers containing the k item set, and the count of occurrences can be obtained.

The ICAMA algorithm uses the idea of MapReduce to improve the two stages of Apriori algorithm. (1) ICAMA proposed a suitable data structure for simplify the number of occurrences of the item set from traversing the entire data set to summing the bit set of the corresponding item. And then discarded the generation process of the candidate to further improve the efficiency of the algorithm[22]. (2) ICAMA make the frequent k–1 item sets stored in Fk–1 are directly connected to the same two types as YAFIM[23]. If they are connectable, their corresponding Bit Sets are summed, and the Bit Set operation and operation are performed. The processing will be terminated while the transaction number of all connected k itemsets is recorded in the Bit Set. Next, it is determined whether the number of transaction numbers in the Bit Set is greater than the minimum support degree. If it is greater than, the connected item set is a frequent k item set, and the result Bit Set is stored as a key value pair in Fk. The first stage start with convert each row of the dataset into multiple (item, TID) key-value pairs by flat Map(), then reduce By Key() to connect the TIDs of the same key into a string and filter the string at the same time using filter() The number of transaction numbers included in the transaction number is less than the minimum support value[24], and then map() is used to construct a string of each TID into a Bit Set, so the first frequent itemsets will be stored in the form of (item, Bit Set) key-value pairs.

The second stage is to obtain k-item sets through the iterative process which is start from k–1 item sets. First, the candidate k item set is obtained from the frequent k–1 item set self-joining and pruning. In order to make the search candidate set faster, YAFIM stores the candidate k item set in the hash tree. Then start the map task, each map task processes several transactions, to obtain all k pairs in the transaction and searches the hash tree and determine whether it is a candidate set. If it is a k candidate set, it is transmitted as a Key (key, 1) key-value pairs, the reducer counts and filters the parts of the frequent k-item set. This is the implementation of the most classic Apriori algorithm, but it is often because the most time-consuming process of generating candidate sets in this process makes the efficiency of the algorithm constrained.

3.2 Time complexity

It is necessary to make assumptions about some values and then express the analysis results in this form in the form of mathematical expressions. Suppose T represents the number of transactions in the data set to be processed, M represents the number of map tasks in the work, and f represents the number of frequent 1 item sets[25]. The asymptotic time complexity of YAFIM is $O(f^2+\frac{T}{M}\times a^2)$; the asymptotic time complexity of ICAMA is $O(\frac{T}{M}\times f)$.

4 Experiment

4.1 Experimental data

The experimental data is the transaction information of agricultural inputs products collected by the Institute for the Control of Agrochemicals in China Pesticide Digital Supervision & Management Platform[26]. Every day, more than 100000 pesticide operators across the country upload their business information to this platform[27], including price, trading location, the varieties of agricultural products inputs and scale of transactions. The supervision platform generates more than 10 million data records per day, so we took one day's data generated from this platform for analysis and testing the performance of the spark-based Apriori algorithm[28].

4.2 Pseudo-code of the experiment

Input:

The Dataset *D*, which stored in HDFS in the form of data blocks. The minimum threshold of supports *min_sup*.

Processing:

1) Get L_1

```
Instans=sc.tectfile(D)
L1=instans.map( _, 1)
instans.map( _, 1).reduceByKey( _+ _ ).filter( _ >min_sup)
```

2) Construct local matrix *G*

```
Matrix G=The initialized matrix of H×(J+2)
foreach (l in L1)
    foreach (t in Di)
        if (l in t)
            G.add(1)
        else
            G.add(0)
```

3) Get local candidate itemsets

```
for (1<k<maxL){
    for (0≤m<maxL){
        count=0
        for (m<n<maxL){
            while (count<k){
                if (G[m][maxL–1]<k)
                    break
                else
                    count++
            }
            Loscal_sup_count = [use "AND" operation on 'kcolum items' of G]
            Cx.add(<kcolumn_items, local_sup_count>)
        }
    }
}
```

4) Calculate global support, get frequent itemsets

```
Gck = Ck.reduceBykey(_ + _).filter(_.2 < min_sup)
L = instans.map(_, gCk).reduceBykey(_ + _).filter(_ > min_sup)
L += ck.reduceBykey(_ + _).filter(_.2 > min_sup).add(kitems, sup_count)
return L
```

Output:

Frequent itemsets *L* derived from data set *D*.

4.3 Holistic description of the experiment

1) Experimental environment

The computer cluster of the experimental platform consists of eight servers. Each server installs two same Linux systems (Ubuntu, Version.12.04) with exception of computing framework, and the computing frameworks they install respectively are Spark + YARN, Spark + Mesos and Hadoop[29].

2) Experimental steps

The Spark based parallel computing is implemented by Mesos. So, it is necessary that set the host and port of the Spark-Mesos before the experiment. Deploying Spark on YARN to deploy Spark frameworks on YARN requires first installation of Maven3.0.4[30] and configuration of its environment variables. Subsequently, Maven is used to compile and package the Spark kernel separately into an independent jar package. Copy the jar package into the other machines in the cluster complete configuration[31,32].

3) Results

The experiment compares among the performances of single machine Apriori algorithm, Hadoop based parallel computing Apriori algorithm and Spark based parallel computing Apriori algorithm on different size data sets. The results are shown in Figures 3 and 4.

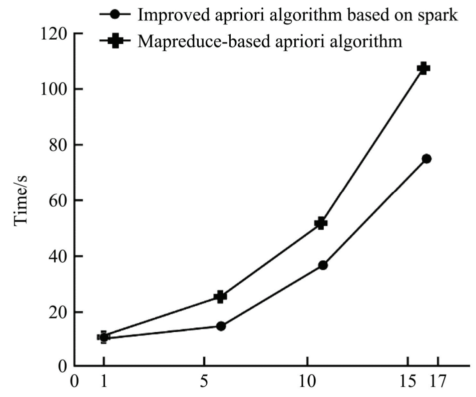

Figure 3 Running time of different data block

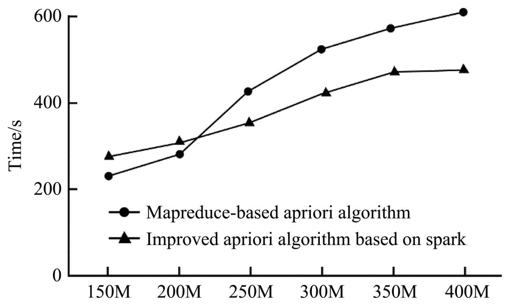

Figure 4 Running time of different algorithm

4.4 Analysis

Experiments show that the scale of dataset to be processed is positively related to computation. So single machine cannot complete Association Rules Mining for large amounts of data limited by computing resources[33]. Although the ICAMA algorithm described in this paper consumes additional running time due to process communication and data transmission, this consumption will not increase greatly due to the expansion of datasets. The larger the amount of data, the smaller the consumption ratio is. Besides, the algorithm has the advantages of parallel computing, such as making full use of computing resources on different machines[34] and reducing the demand for the performance of a single machine[35].

5 Conclusions

This paper briefly summarizes the performance bottlenecks of the classic Apriori algorithm, and improves these aspects, especially the candidate set generation process, and obtains a more optimized algorithm. Then, based on the Spark platform's efficient support for the iterative algorithm, it will improve. The Apriori algorithm is parallelized on Spark and implemented. Then, the detailed analysis and comparison of the existing classic Apriori Spark implementation YAFIM and the improved Apriori algorithm Spark implementation ICAMA are described, and how to improve the algorithm is described. Finally, the efficiency of ICAMA is fully proved theoretically and experimentally. Especially when the amount of data continues to increase, the ICAMA performance improvement will be more obvious. Therefore, the algorithm described in this article has the effecter clustering and has better computational performance on large-scale data. In summary, this algorithm can effectively mine agricultural inputs information, provides the basis for the market regulation of agricultural inputs product markets, and realizes the precise investment of agricultural inputs. And then it provides algorithm basis for achieving the supervision and traceability management of the agricultural inputs market.

Acknowledgements

This work was financially supported by National Natural Science Foundation of China (No. 61601471). The authors gratefully acknowledge the helpful comments and suggestions of the reviewers, which have greatly improved the presentation.

[References]

[1] Li D L. Internet of things and wisdom agriculture. Agricultural Engineering, 2012; 2(1): 1–7. (in Chinese)

[2] Zhao C J. Intelligent agriculture prospects, digital technology will have a new future. Marketing (Agricultural Resources and Markets), 2018; 18: 59–61. (in Chinese)

[3] Zhang M, Jin Y H, Zheng F T. Investigating the "One Farm Household, Two Production Systems" in rural China: The case of vegetable and fruit farmers. Annual Meeting of Agricultural and Applied Economics Association (AAEA), Boston, Massachusetts, July 31-August 2, 2016.

[4] Yan M J, Luo J, Liu J Y, Hou C W. IABS: parallel improved Apriori algorithm based on Spark. Application Research of Computers, 2017; 34(8): 2274–2277.

[5] Zhou Z H, Yang Q. Machine learning and its applications. Beijing: Tsinghua University Press, 2011.

[6] Apache Hadoop. http://hadoop.apache.org/.

[7] Lin J X, Huang Z. An improved Apriori algorithm based on array vectors. Computer Applications and Software, 2011; 28(5): 268–271.

[8] Cao Y, Miao Z G, Zhang H X. Application research about degree warning based on improved Apriori algorithm. Computer Development & Applications, 2014; 27(6): 1–3. (in Chinese)

[9] Zhao X J, Sun Z X, Yuan Y, Chen Y. An improved Apriori algorithm based on orthogonal list storage. Journal of Chinese Computer Systems, 2016; 37(10): 2291–2295. (in Chinese)

[10] Mahout. http://mahout.apache.org/.

[11] Distributed computing. http://baike.baidu.com/view/7011548.htm.

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

[13] Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, April 25-27, 2012.

[14] Mllib. http://spark.apache.org/mllib/.

[15] Low Y C, Gonzalez J, Kyrola A, Bickson D, Guestrin C, Hellerstein J. Graphlab: A new framework for parallel machine learning. https://arxiv.org/ftp/arxiv/papers/1408/1408.2041.pdf.

[16] Berkhin P. A survey of clustering data mining techniques. Grouping multidimensional data. Springer, Berlin Heidelberg, 2006; pp.25–71.

[17] Hu S J. Parallel data mining algorithm research in cloud. Chengdu: University of Electronic Science and Technology of China, 2013. (in Chinese)

[18] Tian S P, Wu W L. Algorithm of automatic gained parameter value k based on dynamic k-means. Computer Engineering and Design, 2011; 32(1): 274–276. (in Chinese)

[19] Suo H G, Wang Y W. Reference-based k-means algorithm for document clustering. Computer Engineering and Design, 2009; 2: 401–403,407. (in Chinese)

[20] He Z, Qian J S. A multicenter clustering algorithm for automatic acquisition of *K* values. Electronics World, 2012; 4: 60–64. (in Chinese)

[21] Bloodgood M, Ye P, Rodrigues P, Zajic D, Doermann D. A random forest system combination approach for error detection in digital dictionaries. Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data. Association for Computational Linguistics, 2012; pp.78–86

[22] Mahdi M U. Determining number and initial seeds of *K*-means clustring using GA. Journal of Babylon University/Pure and Applied Sciences, 2010; 18(3): 1–6.

[23] Lu S L, Lin S M. Distance-based outliers detection and applications. Computer and Digital Engineering, 2004; 32(5): 94–97. (in Chinese)

[24] Granitto P M, Furlanello C, Biasioli F, Gasperi F. Recursive feature elimination with random forest for PTR-MS analysis of agroindustrial products. Chemometrics and Intelligent Laboratory Systems, 2006; 83(2): 83–90.

[25] Zheng L Z, Huang D C. Outliers detection and semi-supervised clustering algorithm based on shared nearest neighbors. Computer Systems and Applications, 2012; 21(2): 117–121. (in Chinese)

[26] Ho T K. The random subspace method for constructing decision forests. EEE Trans. Pattern Analysis and Machine Intelligence, 1998; 20(8): 832–844.

[27] Genuer R, Poggi J M, Tuleau-Malot C. Variable selection using random forests. Pattern Recognition Letters, 2010; 31(14): 2225–2236.

[28] Shin M, Kang E M, Park S H. Automatically finding good clusters with seed *k*-means. Genome Informatics Series, 2003; pp.326-327.

[29] Arthur D, Vassilvitskii S. *K*-means++: The advantages of careful seeding. Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms. Society for Industrial and Applied Mathematics, 2007; pp.1027–1035.

[30] Yong K. Research on feature selection and model optimization of random forest. Harbin: Harbin Institute of Technology, 2008. (in Chinese)

[31] Bahmani B, Moseley B, Vattani A, Kumar R, Vassilvitskii S. Scalable *k*-means++. Proceedings of the VLDB Endowment, 2012; 5(7): 622–633.

[32] Caruana R, Niculescu-Mizil A. An empirical comparison of supervised learning algorithms. Proceedings of the 23rd International Conference on Machine Learning, ACM, 2006; pp.161–168.

[33] Breiman L. Random forests. Machine Learning, 2001; 45(1): 5–32.

[34] Bootstrap. http://en.wikipedia.org/wiki/Bootstrap_aggregating.