Parallel Design of Apriori Algorithm Based on the Method of "Determine Infrequent Items & Remove Infrequent Itemsets"

Suo Dongnan¹, Zhang Zhaopeng¹,a

¹ Datang Northeast Electric Power Test & Research Institute, Changchun, Jilin, 130000, China
²gongyuanyang@neepu.edu.cn
³Corresponding author’s e-mail: suodongnan@163.com

Abstract: In the method of fault association rule diagnosis, Apriori algorithm has low efficiency for big data processing. In this paper, aiming at the defects of Apriori algorithm, MapReduce computing framework is used to optimize the Apriori association rule algorithm. This method improves the accuracy of association mining in fault diagnosis. In the process of optimization, this paper proposes the method of "Determine Infrequent Items & Remove Infrequent Itemsets". Through experiments, this method effectively reduces the computational space needed by Apriori algorithm in association rule mining, and improves the computing speed.

1. Introduction
With the development of electric power company and the improvement of science and technology, the quantity and precision of power plant equipment have been improved continuously. A large amount of data are stored in DCS system of power plant. There are a lot of knowledge and rules hidden behind these data. Now data mining technology has been applied in many aspects in power industry. Data mining technology can extract or mine the characteristics of knowledge from a large number of data, and data mining technology can mine potential rules and knowledge in fault diagnosis. Through these rules and knowledge, people can make correct decisions[1].

Association rule technology is a very important method in the field of data mining. FP-growth and Apriori are classic association rules algorithms, which have important significance in data mining. Through the calculation process of association rules algorithm, fault categories can be found. The relationship with the phenomenon of failure. The paper [2] uses the traditional Apriori algorithm to mine the data of condenser-related faults in power plants, and excavates the relevant laws of several typical fault phenomena in theory. The paper [3] adopts the Apriori algorithm-DLG algorithm with optimized design, through this method, the power grid fault is diagnosed, and the effectiveness of fault diagnosis is improved. The Apriori algorithm has the following shortcomings. First, the selection of minimum support and minimum confidence is based on subjective judgment. Secondly, a large number of candidate sets will be generated in the process of algorithm calculation, which increases the computational complexity and reduces the processing capacity and computational speed of large data.

2. Related Technology Concept Description

2.1. Association Rules
I = \{i_1, i_2, \ldots, i_n\} is a collection of transaction items, where \(i_k (k = 1, 2, \ldots, n)\) is defined as a project, and a project set is defined as a project set \[^4\], and an item set containing \(k\) items is called a \(k\)-item set. D = \{d_1, d_2, \ldots, d_m\} is a transactional database, where \(d_i (i = 1, 2, \ldots, m)\) is called transaction, and each is a collection of items, that is, \(d_i \subseteq I\). Let \(X\) be an itemset, and if and only if \(X \subseteq d_i\), the transaction contains item set \(X\). Association rules are expressed as \(X \Rightarrow Y\), \(X\) as rule precursor, \(Y\) as rule precursor, where \(X \subseteq I, Y \subseteq I,\) and \(X \cap Y = \emptyset\).

Support indicates the probability of the occurrence of a project in a project set. The formula is:

\[
\text{support}(X \Rightarrow Y) = \frac{P(X \cup Y)}{P(X)}
\]

(1)

The confidence level reflects the strength of the association rule \(X \Rightarrow Y\), and the formula is:

\[
\text{confidence}(X \Rightarrow Y) = \frac{P(X \cup Y)}{P(X)}
\]

(2)

The lifting degree indicates whether \(X \Rightarrow Y\) is a valid rule. When it is greater than 1, it is effective. When it is less than 1, it is invalid, equal to 1 when \(X\) and \(Y\) are independent of each other. The calculation formula is:

\[
\text{lift}(X \Rightarrow Y) = \frac{P(X \cup Y)}{P(X)P(Y)}
\]

(3)

2.2. MapReduce

MapReduce \[^5\] is a parallel computing framework designed by Google Labs for parallel processing of large data on multi-state computers. It uses Map function and Reduce function to process massive data in parallel. The Map function parses the input sub-data set and outputs a key value pair \(<\text{key}, \text{value}>\), converts it into another key value pair according to the mapping rules, obtains the intermediate results, then classifies the intermediate results according to the key value, transfers the intermediate results containing the same \(<\text{key}, \text{value}>\) to the same Reduce function, and the Reduce function is negative. Responsibility traverses the key pair \(<\text{key}, \text{List(value)}>\) output by Map function, and outputs the new key pair \(<\text{key}, \text{value}>\) through the calculation of Reduce function to get the final result needed for calculation.

3. Apriori Algorithm Based on MapReduce

Apriori algorithm is an algorithm for mining frequent itemsets proposed by Agrawal and R. Srikant \[^7\] in 1994. It obtains frequent itemsets iteratively and extracts association rules accordingly. Aiming at the shortcomings of traditional Apriori algorithm, this paper applies parallel algorithm in Apriori, optimizes Apriori by MapReduce parallel processing, improves the computing performance of frequent itemsets by determining infrequent itemsets, reduces the number of scans of transaction databases, and improves I/O performance.

The key and difficult point of MapReduce-based Apriori algorithm in programming is the design of Map function and Reduce function. In the design of this method, in the Map function, first define, \(D = \{I : I = \{i_j, \ldots, i_s\} \land I \subseteq t_i \land |I| = s, j \notin L_{inf}, j \subseteq I\}\), where \(s\) is the length of the item set, used to calculate the support degree of the item and the confidence of the frequent items, the data set \(L_{inf}\) is used to store the infrequent items, and the design block pair \(I \in D\) calculates \(\text{supp}(I) = 1/s\), outputs the result of the key value pair in \(<I, \text{supp}(I)>\) format, and sends the key value pair with the same key value to the same Reducer processor. In the Reduce function, first design the block that receives the key-value pairs transmitted by the Map function, and statistically sum the value of the key-value pairs to find the final I support. If \(\text{support} \geq \text{min_supp}\), determine I For frequent itemsets, the processing result is output as a new key-value pair, and the item set that does not satisfy the above relationship is set as an infrequent item set, stored in \(L_{inf}\) and removed from the transaction database, and the frequent item set is calculated. Confidence method to determine that the frequent item set is a strong association rule Then, if \(\text{confidence} \geq \text{min_conf}\), define it as a strong association rule and output it.
The algorithm flow chart is shown in Figure 1. The frequent itemsets are constrained by setting the minimum support degree. The Map function and the Reduce function are designed and compiled. The transactions in the transaction database D are divided into n sub-blocks and assigned to the Mapper processor. Generate a candidate set by sorting the Map function, and generate a key-value pair of < I, supp(I)> format, and send the key-value pairs with the same key value to the same Reducer processor, through n different The Reducer processor integrates the candidate set specifications to generate frequent itemsets, calculates the support and confidence of the frequent itemsets, generates new key-value pairs, and defines items less than the minimum support min_supp and the minimum confidence min_conf as infrequent. Item, and remove the item from the transaction database D, self-join the items in the transaction library D, generate the next-level frequent itemsets, scan the transaction database that removes the infrequent items, repeat the above operations, and cycle back and forth. Until the frequent itemset is empty, end the operation and terminate the program. Compared with the traditional Apriori algorithm, the optimized algorithm considers the minimum support as the basis of the search space pruning to mine the frequent itemsets, and uses the MapReduce method to perform parallel computing. Starting from the mining frequent sequence, the item that does not satisfy the minimum support is defined as Infrequent itemsets, in the next iteration calculation, infrequent items with support below the minimum support will no longer be scanned.

The working principle of the algorithm is shown in Figure 2. In transaction database D, the number of transaction sets is 7 and the minimum support min_supp is 3/7. The first iteration task is to compute frequent 1-item sets. In Map phase, the transaction item sets are randomly assigned to four Mappers for counting and statistics, and the output is < I, the key pairs in supp(I)> format are responsible in Reduce phase. The intermediate results processed in Map phase are allocated to three Reducers for integration calculation. Items with support less than min_supp are sent to Linf. In the first iteration, it is found that the confidence of A3 is less than min_supp. Therefore, A3 is confirmed as a non-frequent itemset and stored in Linf. Therefore, A3 will not be scanned in the next iteration. The task of the second iteration is to compute frequent 2- itemsets, and so on until the frequent set is empty. The
optimized algorithm no longer considers non frequent items, which reduces computation requirements and improves computation speed.

Fig.2 Schematic diagram of algorithm running

4. Experimental analysis
During the experiment, 200,000, 2 million, and 20 million pieces of data will be mined separately, of which 200,000 pieces of data will come from the network, and 2 million pieces of post-experimental experiments will be synthesized based on 200,000 pieces of data from the network and self-compiled data. 20 million pieces of data.

Experiment 1: When selecting a data set containing 200,000 pieces of data, the number of individual items varies from 8 to 20. The original Apriori algorithm and the improved algorithm are used to mine the frequent itemsets of the data set. The experimental results are shown in Fig.3 shows.

Fig.3 Comparison of frequent item set mining time of 200,000 pieces of data

In Fig.3, I-Apriori-MR represents the optimized algorithm of this paper. In the process of mining data sets of 200,000 data, the number of individual items varies from 8 to 20. When the number of individual items is the same, the running time of the original Apriori algorithm mining frequent itemsets is more than that based on the MapReduce optimized algorithm. The time required is shorter; as the number of individual items increases, the time required for both algorithms to run increases exponentially, and the upward trend of the original Apriori is slower than the upward trend of the optimization algorithm; when the number of items increases to After 16 times, the time required for the two algorithms to run is growing faster, and the algorithm optimized by this paper is higher than the original algorithm.
**Experiment 2:** When selecting a data set containing 2 million pieces of data, and the number of individual items varies from 8 to 20, the original Apriori algorithm and the improved algorithm are used to mine the frequent itemsets of the data set. The experimental results are shown in the figure 4.

![Fig.4 Comparison of frequent item set mining time of 2 million pieces of data](image)

Fig. 4 shows that the mining speed of the original algorithm is faster when the number of individual items in the data set is less than 10; when the number of items is more than 12, the time spent by the optimization algorithm in frequent itemsets mining is less than the time required by the original algorithm, and the more the number of individual items, the more obvious the effect is; with the increase of the number of individual items, the two algorithms are mining. When mining frequent items, the time spent increases exponentially, but the growth of the optimization algorithm in this paper is more stable, and the mining speed is faster than the original algorithm.

**Experiment 3:** When selecting data sets containing 20 million data sets, the processing time of the original algorithm is much longer than that of the optimized algorithm in this paper. So this experiment chooses the conventional Apriori algorithm based on MapReduce optimization compared with the Apriori algorithm based on MapReduce optimization proposed in this paper for infrequent itemsets. On the basis of 20 million data, the number of projects varies from 4 to 18. Two algorithms are used to mine frequent itemsets. The experimental results are shown in Figure 5.

![Fig.5 Comparison of frequent item set mining time of 20 million thousand pieces of data](image)

In Figure 5, Apriori-MR represents the conventional MapReduce-based optimization algorithm. From the graph, when the number of individual projects is the same, the running time of the optimized algorithm in this paper is shorter than that of Apriori-MR, and it is more advantageous when the number of projects is small. However, with the increase of the number of individual projects, the running time cost of the optimized algorithm in this paper shows an exponential upward trend, and
when the number of projects increases. To a certain extent, compared with the conventional MapReduce-based optimization algorithm, the time gap between the two algorithms will gradually decrease when generating frequent itemsets.

After the experiment is summarized, the following conclusions are obtained:

- When dealing with small-scale data sets, the Apriori algorithm has advantages over the optimized algorithm in this paper. The stability of data processing is better, the time required for running is shorter, and the running speed is faster.

- When dealing with large-scale data sets, the Apriori algorithm has a small running time when mining frequent items, but the difference between the optimization algorithm and the optimization algorithm is small, negligible; when there are too many individual items The algorithm optimized in this paper is more stable in data processing, and the calculation takes less time and speed.

- When the data size is too large, the Apriori algorithm cannot perform accurate mining. On the basis of MapReduce, the method of removing infrequent itemsets is more advantageous. The running time of the algorithm is relatively short and the running speed is faster, but when the number of items is When it is too large, it is easy to cause too many infrequent itemsets to occur. When the infrequent itemsets are too large, there are still a lot of computing requirements.

5. Conclusion

The research shows that Apriori algorithm is easy to produce a large number of candidate itemsets in the calculation process. Therefore, Apriori algorithm will appear redundancy when processing big data, which will affect the calculation accuracy of the algorithm. In this paper, the parallel design of Apriori algorithm is carried out through MapReduce computing framework, and the speed of Apri algorithm is improved. In the parallel design based on MapReduce, a technical method of "Determine Infrequent Items & Remove Infrequent Itemsets" is proposed, which reduces the generation of candidate item sets, delays the occurrence of redundancy, and solves the need for algorithms in the rule extraction process The problem of multiple scans of the database and heavy I/O load improves the running speed of the algorithm, so that the algorithm can still maintain a high processing speed and calculation accuracy when faced with large-scale data.

References

[1] Su P. Research on the electricity characteristics based on big data[D]. Beijing: North China Electric Power University, 2017.
[2] Liang Z, Chen P, Su H. Association rule mining in fault monitoring of power plant equipment[J]. Electric Power Automation Equipment, 2006, 26(06): 17-19.
[3] Nie Q, Gao W. A power network fault diagnosis method based on data mining association rules[J]. Power System Protection and Control, 2009,37(09): 8-14.
[4] Zhang Y, Hu C, Huang S, et al. Apriori algorithm based data mining and analysis method of secondary device defects[J]. Automation of Electric Power System, 2017, 41(19): 147-151+163.
[5] Mei H, Mi Z, WU G. Massive data processing of intermittent energy based on MapReduce model[J]. Automation of Electric Power System, 2014, 38(15):76-80+99.
[6] Meng X, Liu Y, Gao X, et al. A new bio-inspired algorithm: Chicken swarm optimization[C]//5th International Conference on Swarm Intelligence. Hefei: Springer International Publishing, 2014, 88749: 86-94.
[7] Agrawal R, Srikant R, et al. Fast algorithms for mining association rules [C]. InProc. 20th int. conf. very large data bases, VLDB. 1994: 487-499.
[8] José M L, Francisco P, Mykola P. Apriori versions based on MapReduce for mining frequent patterns on big data[J]. IEEE Transactions on Cybernetics, 2017, PP (99) : 1-15.
[9] Yang S, Xu G, Wang Z, et al. The Parallel Improved Apriori Algorithm Research Based on Spark[C] //9th International Conference on Frontier of Computer Science and Technology. Dalian, 2015: 354-359.
[10] Lin J, Schatz M. Design patterns for efficient graph algorithms in MapReduce[C]//Proceedings of the Eighth Workshop on Mining and learning with Graphs. New York: ACM, 2010:78-85.