Research on Data Mining Algorithm of Association Rules Based on Hadoop

Linrun Qiu (✉)
Department of Computer Science, Guangdong University of Science and Technology, Dongguan, China
79713159@qq.com

Abstract. Big data mining based on cloud computing is the hot topic of the industry research, this paper proposed an improved distributed Apriori algorithm. More importantly, In view of the poor performance of running Apriori algorithm in large data, the algorithm of association rule data mining based on Apriori algorithm is put forward, and the improved distributed Apriori algorithm based on Hadoop platform is proposed. The algorithm focuses on the application of association rules algorithm based on Hadoop in mass data mining. This paper describes the idea of improved Apriori algorithm on Hadoop platform, and presents the experimental test. The experimental results show that the improved algorithm of association rules based on Hadoop can effectively improve the Apriori algorithm for association rules of operation efficiency, and reduce the redundant association rules, and has the efficient advantage in dealing with massive data.

Keywords: Hadoop · Association rules · Apriori algorithm · Data mining

1 Introduction

Data mining is the extraction of implicit, previously unknown and potentially valuable knowledge and rules from large databases or data warehouses. It is the combination of artificial intelligence and database development, in which information decision-making system is in the forefront of research. The main algorithms of data mining are classification pattern, association rules, decision tree, sequence pattern, clustering model analysis, neural network algorithm [1]. The association rules are a very important research topic in the field of data mining. It is widely used in various fields, which can test the long-term knowledge pattern in the industry and discover the new laws. It is an important method to accomplish the data mining task effectively, Therefore, it is significant to study the association rules.

With the rising of cloud computing technology, how to extract knowledge from the massive data is currently an urgent problem to be solved. At the same time, people pay more and more attention to data mining technology in big data environment applications, many scholars made some improved research in the field, such as data classification, clustering and so on. As an important part of data mining, association rule mining plays an important role in data mining. The status of Aprio, a classic algorithm in association rules, is more important [2]. However, the traditional serial algorithm
can’t meet the requirements of big data era. In recent years, cloud computing platform
distributed system infrastructure to make full use of large clusters for high-speed
computing and storage for big data mining provides a parallel computing framework.
Based on the Apriori association rule mining algorithm, this paper proposes an
improved algorithm based on Hadoop framework for distributed parallel mining
algorithm, which can be applied to the mining of large data association rules, and
improve the efficiency of association rule mining in Apriori algorithm under large data
environment.

2 The Technology of Hadoop Platform

Hadoop is an open source parallel computing programming tool and a distributed file
system developed by the Apache Software Foundation that allows users to develop a
highly scalable distributed batch processing system with its MapReduce programming
model without knowing the underlying details [3]. Hadoop’s framework is the core
design of HDFS and MapReduce. HDFS provides storage for massive amounts of data,
and MapReduce provides calculations for massive amounts of data. Hadoop mainly by
the HDFS (Hadoop Distributed File System) and MapReduce engine composed of two
parts. The bottom is HDFS which stores the files on all storage nodes in the Hadoop
cluster. The upper layer of HDFS is the MapReduce engine. The engine consists of
JobTrackers and TaskTrackers. Hadoop is an open source distributed parallel pro-
gramming framework that implements the MapReduce computing model. Program-
ners can use Hadoop to write programs that run programs on a computer cluster to
handle massive amounts of data [4]. With the help of Hadoop framework and cloud
computing core technology MapReduce to achieve data calculation and storage, and
HDFS distributed file system and HBase distributed database is well integrated into the
cloud computing framework, so as to achieve cloud computing distributed, parallel
computing and Storage, and the ability to achieve good processing of large-scale data.
The Hadoop storage architecture is shown in Fig. 1. Particle swarm algorithm.

Fig. 1. The architecture of Hadoop storage
Hadoop uses a master/slave architecture in both distributed and distributed storage. Distributed storage system that Hadoop file system, or referred to as HDFS. NameNode is located on the HDFS master and directs the underlying DataNode to perform underlying I/O tasks. Secondary Name Node (SNN) monitors HDFS cluster status. JobTracker connects applications and Hadoop clusters [5]. TaskTracker manages the operation of each node. MapReduce is Google’s proposed software architecture, but also Hadoop programming model. The data is matched in the MapReduce as a key/value pair:

Map: \((K1, V1) \rightarrow \text{list}(K2, V2)\)

Combiner: \((K2, \text{list}(V2)) \rightarrow \text{list}(K2, V2)\)

Reduce: \((K2, \text{list}(V2)) \rightarrow \text{list}(K3, V3)\).

Combiner can be used as part of the Map function, the Map output of the same key results in the results of the merger, similar to the local Reduce. The purpose of doing so can reduce the data transmission of the network and improve the performance of data processing.

The master service control cluster is equivalent to the controller part, which is responsible for receiving the application request and answering according to the request type. Storage node cluster is equivalent to the memory part, is a huge disk array system or a large amount of data storage capacity of the cluster system, the main function is to deal with data access [6]. Hadoop has a master server (called JobTracker) for scheduling and managing other computers (called TaskTracker), and JobTracker can run on any computer in the cluster. Users do not directly read through the Hadoop architecture and HDFS and HBase access to data, thus avoiding a large number of read operations may cause system congestion. After the user passes the information from the Hadoop architecture to the master service control cluster, the user performs the read operation directly with the storage node. The Hadoop cluster structure is shown in Fig. 2.

![Fig. 2. Structure of Hadoop cluster](image-url)
Spark is a highly fault-tolerant, memory-based distributed cluster computing framework. Relative to the Hadoop cluster computing framework, Spark cluster computing is stored in memory, and Hadoop is stored on the distributed file storage system HDFS (Hadoop Distributed File System). Hadoop computing process will produce a lot of input and output files, so the processing time is longer, you also need to rely on storage resources. Spark’s computations share memory in the form of Resilient Distributed Data sets (RDDs), which are more efficient in iterative or interactive computing. Spark supports more data processing interfaces than Hadoop’s MR (Map and Reduce) interfaces that process data. Spark as a distributed computing framework relies on resource containers, early support for Mesos and Yarn as resource containers, and later also supports stand-alone cluster standalone mode as well as Local mode. Which Yarn mode compatible with Hadoop, Spark is conducive to the deployment of Hadoop based on the deployment of the framework to improve performance.

3 The Data Mining Algorithm of Association Rules

3.1 Association Rules

Association Rules is a logical implication of \( X = \{x_1, x_2, \ldots, x_m\} \rightarrow Y = \{y_1, y_2, \ldots, y_n\} \), where X and Y are called the precedence and succession of association rules, Association rules suggest that there may be strong relationships between the pilot and the followers [7]. Let \( i = \{i_1, i_2, \ldots, i_m\} \) be a set of m different items, given a transaction database \( D \), where each transaction \( T \) is a set of items in \( I \), which is \( TI \), \( T \) There is a unique identifier TID. If the set \( X I \) and \( XT \), then the transaction set \( T \) contains the item set \( X \). An association rule is the implication of the form \( X Y \), where \( XI, YI, X \cap Y = \emptyset \). (1) It has the support degree \( s \), that is, the transaction with at least \( s\% \) in the transaction database \( D \) contains \( X \cup Y \); (2) It has the confidence \( c \), that is, the transaction database \( D \) contains \( X \)’s business has at least \( c\% \) and also contains \( Y \). The association rule mining problem is to find the association rule with the minimum support \( \text{minsup} \) and the minimum confidence \( \text{minconf} \) in the transaction database \( D \) [8]. The association rule mining process consists of two phases:

(1) to find out that there is support for all strong itemsets \( X \) in the transaction database support \( (X) \) is not less than the minimum support \( \text{minsup} \) given by the user, then \( X \) is the large item set.

(2) the use of strong items to generate association rules. For each strong set \( A \), if \( B A \), \( B \neq \), and support \( (A)/\text{support}(B) \geq \text{minconf} \), then the associated rule \( B \) (A–B).

Definition 1. Support: Suppose rule \( x_1 \rightarrow y_1 \), support that \( x_1 \) as a pilot and \( y_1 \) as a successor in the data set at the same time the probability.

Definition 2. Confidence: Suppose rule \( x_1 \rightarrow y_1 \), trust degree represents the number of occurrences of \( x_1 \) as lead and \( y_1 \) as subsequent successions divided by \( x_1 \) as the number of occurrences of the pilot.

Definition 3. Frequent set: the support of the rule is not less than the support threshold, the rule is called the frequent set; otherwise, the non-frequent set.
3.2 Apriori Algorithm

In practice, there are many kinds of association rules mining algorithms, among them, the most classic is Apriori algorithm which is proposed by RAgrawal and RSrikant in 1994 and it is the basic association rules of data mining algorithms. Apriori algorithm for many areas, such as recommended systems, social networks, e-commerce platform. Apriori algorithm uses “in a given transaction database D, any subset of the strong set of items are strong items set; any weak sets of superset are weak items” This principle of the transaction database for multiple scans, the first scan (ie, the large k-1 term set Lk-1) and the function Apriori-gen are used to generate the candidate for the k-1-second scan before the k (k > 1) And then determine the number of support for each element in Ck during the scanning process. Finally, the large k - item set Lk is calculated at the end of each scan. When the candidate k - item set Ck is empty When the end [9]. The frequent item sets of the algorithm are generated as follows:

Input: data set, minimum support threshold minSup, minimum confidence threshold minConf;
Output: association rules;

(1) The data of the data set by line scan, combined with minSup, produce frequent 1-item set, and the initial k = 1;
(2) Under the action of Lk cut and connected to produce a collection of Ck + 1. Combined with minSup, resulting in frequent (k + 1) - item sets;
(3) If Lk + 1 is not empty, then k is incremented by 1, and step (2) is performed again. Otherwise,
(4) Get all the frequent itemsets from step (3). Combined with minConf, generate strong association rules, the algorithm ends.

4 The Improved Apriori Algorithm

The main problems in the practical application of Apriori algorithm are as follows: The more times the database is scanned; the longer the operation time is. each stage of the Ck is too large; can’t be updated, can’t directly deal with numerical data; database. The association rule mining can’t be applied directly; performance and efficiency are low. Therefore, based on the Apriori algorithm, this paper proposes an improved Hadoop platform based distributed association rule mining algorithm.

Spark is a general-purpose, large-scale data processing engine that primarily provides an abstract object based on memory computing RDD, allowing users to load data into memory and use it repeatedly [10]. Spark programming model reference MapReduce, the difference is that Spark memory-based computing features in some applications on the experimental performance more than MapReduce100 times. Spark platform written entirely by the Scala language, Scala is a fusion of object-oriented and functional programming language, Which is specifically designed for distributed, streamlined and concurrency.

The basic operations of RDD include transformation and action. You can construct a new RDD from a Scala collection or a Hadoop data set or generate new RDDs from...
existing RDDs, such as map, filter, groupBy, and reduceBy [11]. Action is calculated by RDD to get one or a group of values, such as count, collect and save. Spark all the conversion is inert, not directly calculate the results, just remember to apply to the basic data set (such as a file) on These conversion actions. Only when a request to return to the results of the Driver action, these conversions will really run, this design allows Spark to run more effectively.

4.1 The Apriori Algorithm Based on Hadoop

The idea of matrix based Apriori algorithm is implemented on Hadoop. The concept of matrix is introduced. It only needs 2 times to scan data set D, combined with Spark technology framework, and improves the efficiency of association rule mining by using local pruning property and global pruning property to improve the process of frequent itemsets generation [12]. Transaction data set and frequent item sets Hadoop-based HDFS file system. The matrix of the behavior of the transaction set, the matrix as a collection of items, vector matrix storage variables of 0 and 1, can reduce the data storage space, reduce the number of scanning, According to the operation rules of the vector, the support of the item set can be generated quickly by using the AND operation in the matrix. According to the Spark internal mechanism, the entire Spark programming framework is based on the operation of RDD [13], the specific algorithm description as follows:

(1) Scan the transaction database, seeking a set of 1 sets of frequent.
(2) The transaction database stored on HDFS is an RDD, RDD is split into n data blocks, and these data blocks are allocated to m work nodes for processing.
(3) Construct a local matrix. Let Di be a data block in the transaction database (1 ≤ i ≤ n). Assume that the number of L1s is H and the number of transactions in Di is J, The H × (J + 2) matrix Gi is constructed by L1 and Di, where the first column is the term in L1, The last column is the support count for the corresponding item in L1, The remaining columns in Gi are the transactions T in Di, If there is an item corresponding to L1 in T, the corresponding position is set to 1, otherwise it is set to 0.
(4) Use Gi to generate local support for candidate item sets, The candidate set containing k items is the set of k items in the first column of Gi, It is only necessary to perform the AND operation on the corresponding k rows in Gi to calculate the local support for the set of candidates containing k items. In the case of a k candidate set, you only need to “OR” with the row of the last column greater than k, Can greatly reduce the number of candidate items. Using local pruning properties, delete the local support degree is less than the threshold of the item.
(5) Use the ReduceByKey operation to get the global support for the candidate set. If the global support is greater than the support threshold, the item is added directly to the frequent item set L; If the global support is less than the support threshold, the global pruning property is used to scan the database again, and finally the set of frequent item sets L.
4.2 The Implementation of Apriori in Hadoop Platform

Transplant the Apriori algorithm to the Hadoop platform based on the Spark programming framework [14], using the Scala language program, the distributed Apriori algorithm to achieve the pseudo-code is as follows:

Input: Data set D (stored as a data block in Hadoop's distributed file system), minimum support threshold min_sup.

Output: The set of frequent itemsets in D.

(1) Solving $L_1$

```scala
instans=sc.textfile(D);
L1=instans.map(_,1).reduceByKey(_+ _).filter(_> min_sup);
```

(2) Construct a local matrix

Matrix $G = \emptyset$; // initialize the $H \times (J + 2)$ matrix

```scala
foreach(l in L1){}
foreach(t in Di){}
if(l in t)
   G.add (1); // If the item l in l is in transaction t, the corresponding position is 1
else
   G.add (0); // otherwise, the corresponding position 0
```

(3) To find the local candidate set

```scala
for (1 < k< maxL ) {//maxL is the number of rows of matrix G
   for (0 = <m <maxL) {
      Count = 0;
      for (m <n <maxL) {
         while (count <k) {// count count, make sure k lines do "with" operations
            // The last column value is less than k, no "AND" operation
            if (G [m] [maxL-1] <k)
               break;
            else
               Count += 1; }
         Local_sup_count = [use "AND" operation on 'k column items'of G];
         Ck.add (<k column items, local_sup_count>);
      }
   }
}
```

(4) calculate the global support, get frequent item sets L

```scala
/* Apply global pruning policies to items with global support less than minimum support, traverse transaction databases for pruning */
GCk = Ck.reduceByKey (_ + _). Filter (_< min_sup);
// items with global support greater than minimum support are added directly to frequent itemsets
L = instans.map (_, gcK) .reduceBykey (_ + _). Filter (_> min_sup) ;
L += Ck.reduceBykey (_ + _). Filter (_> min_sup) .add (k items, sup_count);
return L;
```
5 Experimental Results and Analysis

In order to verify the performance of the algorithm, Six nodes in the cluster environment are deployed in the laboratory LAN. the software list as follows: operating system centos 7.2, Java version jre-8.0, Hadoop version hadoop-2.7.2, using Eclipse neon configured MapReduce distribution Programming environment. The test data is based on scrapy crawling the day cat electric business platform in the first quarter of 2017 clothing category user shopping dataset. According to the distributed Apriori algorithm, the Map function, Combiner function and Reduce function are written, and the distributed Apriori algorithm is compared with the traditional Apriori algorithm. The comparison results are shown in Figs. 3 and 4, where the abscissa indicates the amount of data and the ordinate represents the calculation time.

Fig. 3. Apriori algorithm and distributed Apriori algorithm comparison

Fig. 4. Distributed Apriori algorithm comparison of different nodes
Compared with the experimental data, the distributed Apriori algorithm has no obvious advantage, but with the increase of the data, the time of the two algorithms is linearly increasing. When the amount of data grows to 350,000, the advantages of the distributed Apriori algorithm begin to appear. At this time, the Apriori algorithm can’t continue to run because of insufficient memory, and the distributed Apriori algorithm can continue to run and the slope of the time change becomes slow. So as the amount of data increases, the operation time slows down.

Figure 4 Comparison of experimental data Note: In the case of the same amount of data, DataNode nodes increase, you can reduce the algorithm’s computing time. In theory, when the number of nodes doubled, the operation time should be halved, but the actual situation is not the case. The reason why the operation log analysis is combined with the system may be: in the reduce stage of the algorithm, each node needs to calculate the support degree of each candidate set, different calculation amount of different candidate sets and different number of frequent item sets. Each node has a different running time. But on the whole, increase the number of nodes can effectively reduce the system running time.

6 Conclusion

In this paper, an improved mining algorithm of association rules is proposed based on Apriori algorithm. And the implementation of Apriori algorithm is described in Hadoop platform, which proves the advantages of the distributed association rule algorithm in dealing with big data. With the rapid growth of cloud data, the traditional data classification algorithm based on statistics and machine learning method in the handling of big, heterogeneous and complex Web data, the use of ordinary association rules algorithm on the server test is very serious, based on The Hadoop platform’s distributed association rule algorithm can be used to effectively allocate these tests on a cluster machine. Experimental results show that the algorithm is simple and easy to implement, and can effectively improve the efficiency of association rules mining.

Acknowledgements. The work is supported in part by Department of Education of Guangdong Province under Grant 2015KQNCX188.

References