MR-Apriori: Association Rules Algorithm Based on MapReduce

Pooja Reddy Nathala, Srujan Pothina, Pengy Ni
Tseeye Odugbu Potters, Anuj Patel
Department of Computer Science University of North Carolina at Charlotte
Charlotte, NC, USA
e-mail: {pnathala, spothina ,pni1, todugbap, apate139 *}@uncc.edu*

Abstract: Data mining deals with the extraction of patterns and establishment of relationships to solve problems through data analysis. Discovering the beneficial actions from the dataset for the usefulness of decision maker is main task of data mining. Extraction of Action Rules, which accomplishes the task of discovering beneficial actions is attracting a significant amount of research, industry, and media attention of late. There are several methods to extract the action rules and association rules but as the data becomes huge there is a need to introduce scalable methods which can deal with massive datasets. In this paper a powerful technique called Apriori Algorithm implementation in Hadoop MapReduce is explained. MapReduce is a parallel programming paradigm that runs on Hadoop frameworks to provide scalability and easy data-processing solutions. This mapreduce implementation reduces the run time in comparision with Traditional single machine method implementation.

Keywords- MapReduce, Apriori algorithm, Hadoop, Rules

1.Introduction

The need for advancement in data storing and analytical technology has been raised as amount of data is moving from terabytes to petabytes. The prominent data computing model called cloud computing is developed which is the next level of distributed computing, parallel computing, and grid computing. In cloud computing, the job tasks are distributed among nodes in a cluster of computers. These cloud computing techniques along with data mining methods, increases efficiency and capacity.

Discovering frequent patterns in transactional databases is gaining interest in recent years. An action rule is a rule extracted from a database that describes a possible transition of objects from one state to another with respect to a distinguished attribute called a decision attribute. Mainly, there are two types of attributes namely stable and flexible attribute. Depending on the flexible from and To attributes action rules are derived in the advantage of user. For example, given a list of items in a store, the combination of items which need to be sold together to gain maximum profit is derived from action rules and association rules. These rules execution takes much time if they are implemented on huge datasets on a single machine. So, there is a need for parallelization of tasks so that it reduces the time, which is implemented using MapReduce model.

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values with the same intermediate key. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The runtime system takes care of the details of partitioning the input data, scheduling the programs execution across a set of machines, handling machine failures, and managing the required inter-machine communication.

The main use of association analysis of transaction data is in market basket data. There are two main issues that need to be spotted when applying association analysis to market basket data. Primarily, discovering patterns from a large transaction data set can be computationally expensive. Secondarily, some of the discovered patterns are potentially spurious because they may happen simply by chance. These two issues can be solved by discovering algorithm which runs in less time and works accurately. These algorithms generate rules which have properties like support and confidence. Support is the number of transactions that contain a particular itemset or simply the number of times it occurred in a database. Confidence is how frequently items in Y appear in transactions that contain X. According, to these properties values itemset is differentiated as either frequent or non-frequent itemset.

The remaining paper is organized as follows: Section 2 describes related work. Section 3 describes the legacy Apriori algorithm and Apriori Map/Reduce algorithm. Section 4 is Experiments and Results. Section 5 is conclusion.

2. Related Work

The concept of action rules was first introduced by Z.W. Ras and A. Weiczorkowska in 2000[1]. This was followed by various studies where action rules were generated from classification rules which were initially generated by using algorithms such a LERS and ERID[2],[3],[4],[5],[6],

[7],[8],[9],[10]. This studies all employed a classification based action rule mining technique. Z. He et al, (2005) in their paper "Mining action rules from scratch", argued that if action rules were not mined from scratch some meaning action rules will be missed in such classification-based techniques. to mine action rules from scratch to avoid loss of some developed[11]. Their formulation was different from previous work in that it explicitly stated action rules as a search problem in a support-confidence cost framework and provided guarantee on verifying completeness and correctness of discovered action rules[11]. This was done with an algorithm like the Alpriori algorithm.

All the above stated algorithms were not designed to work with large dataset. With the increase in size of data generated and used in various industries in recent times, there was the need for the application of this algorithms in processing these large datasets without taking longer computational time. To improve the computational time for processing large dataset, X. Lin (2014) proposed the use of association rules algorithm based on MapReduce programming model (MR-Apriori) on the Hadoop distributed computing environment and this algorithm worked effectively[12]. Similar model was then used with the action rules algorithm in later years. A. A Tzacheva et al (2016), in their study proposed the adaptation of association action rules mining algorithm for processing large datasets in distributed network using the MapReduce framework (developed by Google in 2005) so as to enhance the computational time and scalability of the action rule algorithm[13]. This work showed that action rules algorithm performed effectively on the Hadoop cluster computing environment with a faster computational time compared to traditional action rules discovery using a single machine.

3. Method

3.1 Legacy Apriori Algorithm

Apriori is association rule learning and frequent item set mining algorithm based used for transactional databases and market basket analysis. It identifies the frequent items set by extend the seed frequent item set one item at a time then stop if no items found. Then action rules can be determined by mining the frequent items sets. Then we can use these action rules to mining the association trends in the database.

Legacy Apriori Algorithm:

```
Aprior (T, threshold):
    L1 ← {large 1 - itemsets}
    k ← 2
    while L(k-1) not emptyset
        C(k) ← {a union b | a belongs L(k-1) ^ b not belong a} - {c| s belong c ^ |s| = k-1 not belong
L(k-1)}
        for transaction t in T
            C(t) ← {c| c belong C(k) ^ C belong t}
            for candidates c in C(t)
                count[c] ← count[c] + 1
        L(k) ← {c|c in C(k) intersect count[c] greater or equal threshold}
        k ← k + 1
    return UkL(k)
```

T is Transaction database, threshold is support threshold, L, C(k) is candidate set for level k, c is candidate set. k is level.

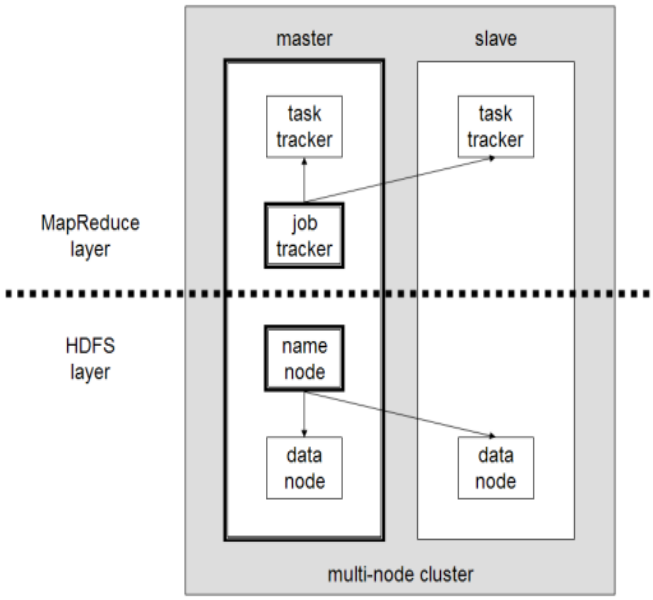
The algorithm is not suitable for big data, since it need to store all the possible frequent items sets and all the candidate rules in the memory, so we proposed a Map/Reduce based apriori algorithm.

3.2 Apriori Map/Reduce algorithm

In the algorithm, we largely depended on the mapreduce framework which can be used deal with big data computation based on the Map and Reduce strategy.

The hadoop is used for processing big dataset based on the MapReduce framework. it is distributed system for storing dataset and process dataset. It includes four parts, (1) Hadoop Common, which hold all the useful libraries for other modules. (2) Hadoop Distributed File Systems (HDFS), which is a distributed dataset storage system, and can provide aggregate utilities for all nodes in the systems. (3) Hadoop YARN, which is a platform for managing computing resources and then scheduling users' jobs. (4) Hadoop Mapreduce: it is computing framework for big data processing.

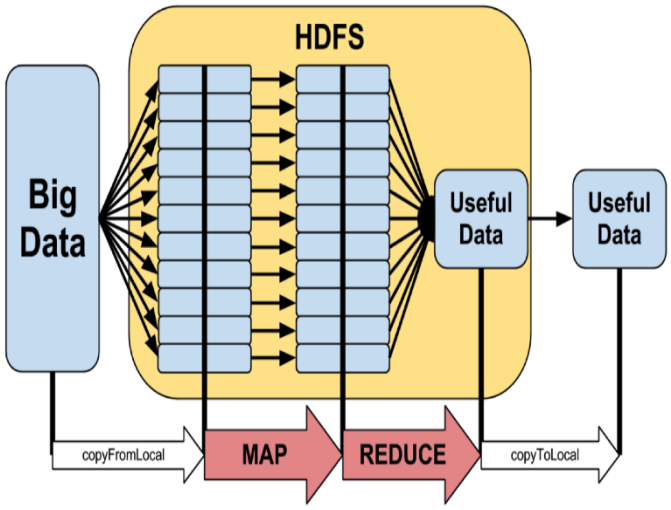
Fig 1. Hadoop Systems.



Hadoop System includes two part: 1) HDFS, it can process the dataset and distribute the dataset onto node in a parallely and locally manner the 2) Process Computing Layer, it process the patches on the local node.

In the hadoop distributed file system, the big data have been split to small pieces, and then all these pieces of file will be distributed to multiple nodes with the code, and then all pieces dataset will be processed in a parallel manner, then the system will collect the map results for all pieces of data and reduce the results to a useful dataset, and finally we can copytoLocal the files on the local host.

Fig 2. Details of HDFS



Pipeline for big data: 1) split big data into patches, 2) copy to each node of the HDFS, 3) Map patches based on <Key, Value> format, 4) Reduce and collect results into files, 4) copy results from HDFS to Local.

In the Map/Reduce computation framework, it includes three parts, (1) split all the files based on the HashMap (key, value), then sort the HashMap by key, then merge the results based on the key. Then output results to the HDFS and then the results can be copyToLocal to local host.

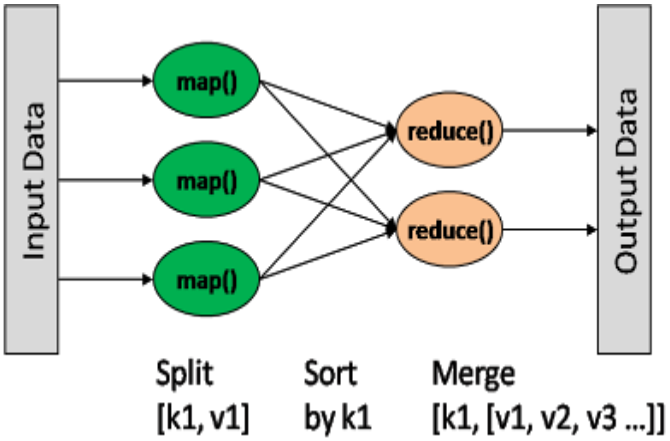


Fig 3. Map/Reduce computation paradigm

In the MR apriori algorithms, the pseudo code as follows:

1. read “file.attribute” and “file.data”, distributed files into hadoop nodes.
2. Map step for key and value.
3. run apriori algorithms for all pieces of data
4. Reduce step for results obtained from all nodes.
5. Output action rules

Since the dataset have been distributed to all the nodes on the cluster, and the most time consuming step will be running parallely, so MR will save lots of time for apriori.

4. Experiment and Result

In experimentation with the code written we are using two different datasets namely Mammographic and Car data set. These datasets are downloaded from UCI Machine Learning Repository. Each data set have two text files: 1.attributes file 2. data file.

The mammographic data set attribute file consists of six attributes namely: birads, age, shape, margin, density

and severity. In the data file we have values corresponding to those attributes. The data file which is considered is csv file (comma separated file) so the code is written considering that. In the data file we can find some special character '?' which means that the attribute value is missing, so we are ignoring those line of data.

Coming to code execution, we executed the code on both cloudera and UNCC DSBA cluster.
*Cloudera Execution Description***:** The input datasets are copied to the hadoop cluster. using these commands:
$hadoop fs -put Car (For Car Data)
$hadoop fs -put Mammographic (For Mammographic data).
The mapreduce implementation of algorithm is written in three java files namely Apriori.java , ActionRules.java and AssociationActionRules.java.

Initial SetUp: Before compiling the java files we need to make a directory 'build' to store all class files generated. We can do this using this command:
$ mkdir -p build.
Now, these three java files are compiled using the following command:
$ javac -cp /usr/lib/hadoop/*:/usr/lib/hadoop-mapreduce/* *.java -d build -Xlint:none .
Next we need to build a jar file using the build directory, this can be done with:
$ jar -cvf Apriori.jar -C build/ .
Now copy the jar file on the local system to the hadoop file system using this command
$hadoop fs -put Apriori.jar.

*Code Execution***:** Now the jar file is available on hadoop cluster so we can run the jar file on the datasets considered. The program code generates two output files ActionRules and AssociationRules. These rules are useful to find the association between products which helps to increase the profits.
Command for Mammographic Data set:
$hadoop jar Apriori.jar apriori.group5.Apriori Mammographic/mammographic_attribute.txt Mammographic/mammographic_masses.data.txt ActionRulesOutput AssociationRulesOutput
Command for Car Data set:
$hadoop jar Apriori.jar apriori.group5.Apriori Car/car.c45-names.attributes.txt Car/car.data.txt ActionRulesOutput AssociationRulesOutput

DSBA Cluster Execution: University of North Carolina at Charlotte have designed a DSBA cluster with 72 nodes for the usage of students. Hadoop blocks in this cluster have a default size of 64 MB. This setting is remained unchanged for our program execution.

Upto building a jar file all steps are common to the DSBA cluster execution. We just need to copy the jar file from local directory to the dsba cluster using this command
$scp path_to_file username@dsba-hadoop.uncc.edu:~/ .

Code Execution:Now the jar file is available on DSBA UNCC cluster so we can run the jar file on the datasets considered. The program code generates two output files ActionRules and AssociationRules.

Command for Mammographic Data set:
$hadoop jar Apriori.jar apriori.group5.Apriori Mammographic/mammographic_attribute.txt Mammographic/mammographic_masses.data.txt ActionRulesOutput AssociationRulesOutput

Command for Car Data set:
$hadoop jar Apriori.jar apriori.group5.Apriori Car/car.c45-names.attributes.txt Car/car.data.txt ActionRulesOutput AssociationRulesOutput

We have the Time Comparision Table which compares the execution time on two different clusters.This time is different as the number of nodes handling the tasks assigned are different.

Table 1. Time Comparison

	# of Node	Time
Apriori on one cloudera	1	~5min
Apriori on Hadoop Cluster	72	~2min

We are presenting the output files generated using Mammographic data set for a particular field values.
Stable Attribute- age
Decision Attribute- shape,
Decision From-shape1
Decision To- shape2
support-2
confidence -20%

Here we are presenting the ActionRules output samples:

Table 2. Action Rules samples

(age,age24 -> age24)^(density,density3 -> density2)^(severity, -> severity0)^(margin, -> margin1) ==> (shape,shape1 -> shape2) 1 [Support: 2.0, Confidence: 100.0%]

(age,age24 -> age24)^(density,density3 -> density2)^(severity, -> severity0)^(margin, -> margin1)^(birads, -> birads4) ==> (shape,shape1 -> shape2) 1 [Support: 2.0, Confidence: 100.0%]
(age,age24 -> age24)^(margin,margin1 -> margin1)^(birads,birads4 -> birads4)^(density,density3 -> density2) ==> (shape,shape1 -> shape2) 1 [Support: 2.0, Confidence: 100.0%]
(age,age24 -> age24)^(margin,margin1 -> margin1)^(birads,birads4 -> birads4)^(density,density3 -> density2)^(severity, -> severity0) ==> (shape,shape1 -> shape2) 1 [Support: 2.0, Conf idence: 100.0%]
(age,age36 -> age36)^(margin,margin? -> margin3)^(birads, -> birads4) ==> (shape,shape1 -> shape2) 1 [Support: 2.0, Confidence: 100.0%] (shape,shape1 -> shape2) 1 [Support: 2.0, Confidence: 100.0%]
(age,age45 -> age45)^(margin,margin2 -> margin1)^(density,density3 -> density?)^(severity, -> severity0)^(birads, -> birads4) ==> (shape,shape1 -> shape2) 1 [Support: 2.0, Confidence: 100.0%]
(age,age45 -> age45)^(margin,margin2 -> margin1)^(density,density3 -> density?)^(severity, -> severity0) ==> (shape,shape1 -> shape2) 1 [Support: 2.0, Confidence: 100.0%]
(age,age45 -> age45)^(margin,margin2 -> margin1)^(density,density3 -> density?)^(birads, -> birads4) ==> (shape,shape1 -> shape2) 1 [Support: 2.0, Confidence: 100.0%]
(age,age45 -> age45)^(margin,margin2 -> margin1)^(density,density3 -> density?) ==> (shape,shape1 -> shape2) 1 [Support: 2.0, Confidence: 100.0%]
(age,age45 -> age45)^(margin,margin2 -> margin1)^(density, -> density?)^(severity, -> severity0)^(birads, -> birads4) ==> (shape,shape1 -> shape2) 1 [Support: 2.0, Confidence: 100.0%]
(age,age45 -> age45)^(margin,margin2 -> margin1)^(density, -> density?)^(severity, -> severity0) ==> (shape,shape1 -> shape2) 1 [Support: 2.0, Confidence: 100.0%]
(age,age45 -> age45)^(margin,margin2 -> margin1)^(density, -> density?)^(birads, -> birads4) ==> (shape,shape1 -> shape2) 1 [Support: 2.0, Confidence: 100.0%]

(age,age45 -> age45)^(margin,margin2 -> margin1)^(density, -> density?) ==> (shape,shape1 -> shape2) 1 [Support: 2.0, Confidence: 100.0%]

Here we are presenting the AssociationRules output samples:

Table 3. Association Rules

(age,age36 -> age36)^(severity,severity0 - > severity0)^(margin,margin1 -> margin3) ==> (shape,shape1 -> shape2) 2.0,50.0 <----> Good
(birads,birads2 -> birads3)^(density,density? -> density3) ==> (shape,shape1 - > shape2) 2.0,26.666666666666668 <----> Good
(age,age37 -> age37)^(severity,severity0 -> severity0) ==> (shape,shape1 -> shape2) 2.0,24.0 <----> Good
(age,age37 -> age37)^(severity,severity0 - > severity0)^(birads,birads4 -> birads4) ==> (shape,shape1 -> shape2) 2.0,24.0 <----> Good
(age,age37 -> age37)^(severity,severity0 - > severity0)^(birads,birads4 -> birads4)^(density,density3 -> density3) ==> (shape,shape1 -> shape2) 2.0,25.0 <---- > Good
(age,age37 -> age37)^(severity,severity0 - > severity0)^(birads,birads4 -> birads4)^(margin,margin1 -> margin1) ==> (shape,shape1 -> shape2) 2.0,25.0 <----> Good
(age,age37 -> age37)^(severity,severity0 - > severity0)^(density,density3 -> density3) ==> (shape,shape1 -> shape2) 2.0,25.0 <----> Good
(age,age37 -> age37)^(severity,severity0 - > severity0)^(margin,margin1 -> margin1) ==> (shape,shape1 -> shape2) 2.0,25.0 <----> Good
(birads,birads2 -> birads3)^(severity,severity0 - > severity0)^(margin,margin1 -> margin1) ==> (shape,shape1 -> shape2) 2.0,44.44444444444444 <---- > Good
(birads,birads2 -> birads3)^(density,density? - > density?)^(margin,margin1 -> margin1) ==> (shape,shape1 -> shape2) 2.0,100.0 <----> Good

5.Conclusion

In this report we studied action rule and association rule mining algorithm which is written in MapReduce programming in a cloud computing platform. Apriori algorithm is applied on the transactional database. By using concept of of apriori algorithm, we generate frequent itemsets for the data. Apriori algorithm is associated with certain limitations of large database scans. Advantage of apriori is its ease of implementation. We have implemented MR-Apriori algorithm on a hadoop platform. Compared the results running the Algorithm on two different hadoop platforms namely Cloudera and DSBA cluster showing which was more better in terms of run time.

Further we can do analysis for the financial data along with the dataset, which we have discussed before like Mammographic and Car. This analysis helps in finding with whom we can take the loan risk, and how to increase the allegiance of customer. We can use this analysis in medical data where it suggests the techniques of curing diseases.

Future work includes implementation of Apriori algorithm in Spark framework. We can also work on the implementation of these algorithms in real time for medical data, financial data and social data.

6. References

[1] Z.W. Ras and A. Wieczorkowska, “Action-Rules: How to increase profit of a company”, in Principles of Data Mining and Knowledge Discovery, Proceedings of PKDD 2000, Lyon, France, LNAI, No. 1910, Springer, 2000, pp. 587-592.

[2] J.Dean and S. Ghemawat, “MapReduce: Simplified Data processing on large clusters” in Proceedings of the 6th conference on Symposium on Operating Systems Design and Implementation. Volume 6, ser. OSDI, 04. Berkeley, CA, USA, USENIX Association, 2004, pp.10-10.

[3] Z.W. Ras, A. Tzacheva, L.S. Tsay, and O. Gurdal, “Mining for interesting action rules”, in Proceedings of IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2005), Compiegne University of Technology, France, 2005, pp. 187- 193.

[4] L.S. Tsay and Z.W. Ras, “Action rules discovery system DEAR3”, in Foundations of Intelligent Systems, Proceedings of ISMIS 2006, Bari, Italy, LNAI, No. 4203, Springer, 2006, pp. 483-492.

[5] Z.W. Ras and A. Dardzinska, “Action rules discovery, a new simplified strategy”, Foundations of Intelligent Systems, LNAI, No. 4203, Springer, 2006, pp. 445-453.

[6] A.A. Tzacheva and Z.W. Ras, “Constraint based action rule discovery with single classification rules”, in Proceedings of the Joint Rough Sets Symposium (JRS07), LNAI, Vol. 4482, Springer, 2007, pp. 322- 329.

[7] Z. Ras, E. Wyrzykowska, and H. Wasyluk, “ARAS: Action rules discovery based on agglomerative strategy”, in Mining Complex Data, Post-Proceedings of 2007 ECML/PKDD Third International Workshop (MCD 2007), LNAI, Vol. 4944, Springer, 2008, pp. 196- 208.

[8] S. Greco, B. Matarazzo, N. Pappalardo, and R. Slowinski, “Measuring expected effects of interventions based on decision rules”, J. Exp. Theor. Artif. Intell., Vol. 17, No. 1-2, 2005, pp. 103- 118.

[9] Y. Qiao, K. Zhong, H.-A. Wang, and X. Li, “Developing event condition-action rules in real-time active database”, in Proceedings of the 2007 ACM symposium on Applied computing, ACM, New York, 2007, pp. 511-516.

[10] J. Grzymala-Busse, “A new version of the rule induction system” LERS, Fundamenta Informaticae, Vol.31, No. 1, 1997, pp. 27-39.

[11] A. Dardzinska and Z. Ras, “Extracting rules from incomplete decision systems”, Foundations and Novel Approaches in Data Mining, Studies in Computational Intelligence, Springer, Vol. 9, 2006, pp. 143-154.

[13] S. Im and Z.W. Ras, “Action rule extraction from a decision table: ARED”. Foundations of Intelligent Systems, Proceedings of ISMIS 08, A. An et al. (Eds.), Springer, LNCS, Vol. 4994, 2008, pp. 160- 168. [14] Z. Pawlak, “Information systems - theoretical foundations”, Information Systems Journal, Elsevier, Vol. 6, 1981, pp. 205-218.