**MR – Random Forest Algorithm for Action Rules Discovery**

Naga Manikanta Sri Venkata Jonnalagadda, Abhilash Gandham, Amruth Sai Gandavarapu, Karthik Gorijavolu

Department of Computer Science

University of North Carolina at Charlotte

Charlotte, NC, USA

email – [njonnala@uncc.edu](mailto:njonnala@uncc.edu), [agandham@uncc.edu](mailto:agandham@uncc.edu), [agandava@uncc.edu](mailto:agandava@uncc.edu), [kgorijav@uncc.edu](mailto:kgorijav@uncc.edu)

***Abstract* – Action rules are one of the latest approaches for discovering patterns in data. These are slightly modified version of classification rules. Action rules are used to generate different suggestions in approaching data. Through these rules we can classify one class to another class and observe the patterns with respect to other attributes accordingly. We mainly use LERS algorithm to produce these action rules.**

**Action rules can be mainly used in medical field which can also be further extended to other fields. Due to the large scale increase in data in every field these action rules can be used to observe different patterns across the data, which further increases the chance of obtaining better results. In this paper we are going to explain how action rules are generated using LERS algorithm and how they can be implemented on Hadoop MapReduce environment. We also explained how association action rules are generated in Hadoop environment. In this paper, we are going to use random forest algorithm to obtain the best rule set among all the rules generated across the given data.**

I. INTRODUCTION

Methods for knowledge discovery in databases have been studied for more than a decade. New methods are required owing to the size and complexity of data collections in administration, business and science. They include procedures for data query and extraction, for data cleaning, data analysis, and methods of knowledge representation [1]. Knowledge discovery is a process that extracts implicit, potentially useful or previously unknown information from the data [2]. Before doing something with the data we have first we have to refine the data. There are several steps in data refining. First step is data cleaning, in which we pre-process the data to remove noise and handle the missing values. The next step is relevance analysis, in which we remove irrelevant and redundant attributes. The final step is data transformation, in which we generalize and normalize the data [3].

Data mining is a process of tuning the knowledge we have and discovering unique patterns present in huge data. Regression, classification, association and clustering are some of the machine learning algorithms used for data mining. Performing these algorithms on huge data sets is a tedious task.  Some of the algorithms that have been developed recently have proven to be easy to be implemented. Some of these algorithms give result as action rules which are generate actions on the discovered patterns. In the features present in the given dataset there is a distinguished attribute (feature) called as decision attribute. Based on the decision attribute action rules can be generated by discovering possible changes of objects from one class to another.  Other than decision attribute there are two more attributes that helps in discovering action rule, they are called as flexible attributes and stable attributes. The attributes which don’t change at all are called stable attributes whereas flexible attributes are the attributes which change from one class to another.

Data mining has been used intensively and extensively by many organizations. In healthcare, data mining is becoming increasingly popular, if not increasingly essential. Data mining applications can greatly benefit all parties involved in the healthcare industry. For example, data mining can help healthcare insurers detect fraud and abuse, healthcare organizations make customer relationship management decisions, physicians identify effective treatments and best practices, and patients receive better and more affordable healthcare services [4].

II. RELATED WORK

2.1 Word count

A word count algorithm is a simple way to count how many time the words appear in the given input. The input given can be a single text file or a collection of multiple text files. We run word count program on Hadoop. There are several real-world applications of word count algorithm. Ranking the documents is one of examples. Consider the following example. Given a certain query there are too many documents related to the query and world we like to rank them according to their importance. Each document’s importance can be calculated by counting the words that appear in the query. If the words in query appear in a certain document more number of time, then that certain document has higher importance that the other documents. So, we give a higher rank to that document. In some cases, the importance of a documents can also be calculated by counting the number of words in a document. As user generated content grows in prominence, many web sites have employed complex mechanisms to help visitors to identify high quality content [5]. It is an arduous word to separate important articles from worthless articles. This process can be automated. By counting the number of words in an article, if the number of words exceed a certain value then that article is more likely to an important article. So, by using word count algorithm if an article has more than a certain number of words then it is assumed to be an important article.

2.2 LERS

Frequently rules induced from raw data are used for classification of unseen, testing data. In the simplest form of classification, if more than one concept was indicated by rules for a given example, the classification of the example was counted as an error. Likewise, if an example was not completely classified by any of rules, it was considered an error. This classification scheme is said to be naive LERS classification scheme [3]. Bases on classical rough set approximations, the LERS data mining system induces two types of rules, namely, certain rules from lower approximations and possible rules from upper approximations. By relaxing the stringent requirements of the classical rough sets, one can obtain probabilistic approximations. The LERS can be easily applied to induce probabilistic positive and boundary rules from probabilistic positive and boundary regions [4]. Learning from Examples based on Rough Sets commonly abbreviated as LERS is used to generate action rules. It is as of now can be used in medical field to see what affects the right sided attribute which is the decision attribute. It is used to observe when a change in the value of decision attribute occurs how are other attributes going to respond.

2.3 Random Forest

Recently there has been a lot of interest in ensemble learning (methods that generate many classifiers and aggregate their results). Two well-known methods are boosting and bagging of classification trees [6]. In boosting, successive trees give extra weight to points incorrectly predicted by earlier predictors. In

the end, a weighted vote is taken for prediction. In

**LERS (attributesSupport, decisionSupport)**

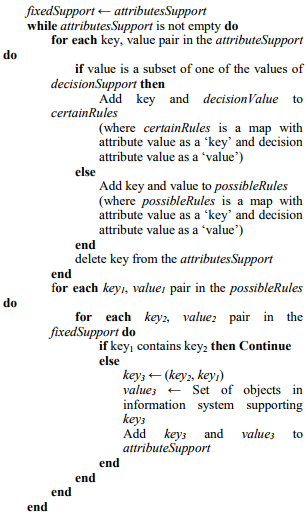


Figure 2. LERS (Learning from Examples based on Rough Sets) Algorithm in a distributed environment using MapReduce [11]

bagging, successive trees do not depend on earlier trees each is independently constructed using a bootstrap sample of the data set. In the end, a simple majority vote is taken for prediction.

Random forests, which add an additional layer of randomness to bagging. In addition to constructing each tree using a different bootstrap sample of the data, random forests change how the classification or regression trees are constructed. In standard trees, each node is split using the best split among all variables. In a random forest, each node is split using the best among a subset of predictors randomly chosen at that node. This somewhat counterintuitive strategy turns out to perform very well compared to many other classifiers, including discriminant analysis, support vector machines and neural networks, and is robust against overfitting.

Random forest is one of the advanced supervised machine algorithms. We can infer from its name that is creates a forest of random decision trees. There is a direct relationship between the number of trees in the forest and the accuracy of the result we finally get. if there are more number of trees then is higher chance the accuracy of the result we get is also high. But creating a forest of decision trees is not same as construction the decision with information gain or gain index approach. The decision tree is decision support tool. It uses a tree like graph to show the possible consequences. If you give a dataset as input with targets and features into the decision tree, then it will formulate certain set of rules. These rules can be used to perform predictions. All the predictions are made by calculating information gain and gini index. The main difference between Random forest algorithm and decision tree is that in random forest, the process of finding the root node and splitting the feature nodes will run randomly. One of the real-world application of random forest algorithm is that given the medical history of a patient, the kind of disease the patient is suffering can be predicted.

III. METHOD

3.1 Word count

We ran word count program on Hadoop. The following are the steps how word count is executed.

  Splitting – Each word in the document must be separated from other words in the document. Splitting can by assigning delimiter as space, comma, semicolon or a new line.

Mapping – It takes a set of data and converts it into another set of data. In the case of word count it takes each word assigns the value 1 for each word.

Intermediate splitting – This entire process is done parallel on different clusters.  Same keys are grouped together so proper execution can be done in reduce phase.

Reduce – All the values of the key are added together so that count of each word can be determined.

  Combine – This is the last phase where all the data is combined to form a result.

LERS

In each dataset we select a decision attribute. For example, let us say we select attribute A as decision attribute with 3 number of classes A1, A2 and A3.

We classify how many rows fall into A1 category and similarly with A2 and A3.

First loop:

We shall classify similarly the other attributes available. Apart from decision attribute there are two types of attributes available.

Stable Attribute: Attribute in which value can’t be classified to another i.e.., the value can’t be changed under meaningful conditions is called stable attribute.

Flexible Attribute: Attribute in which we can classify the value from one to another is called flexible attribute.

By deciding the decision attribute, stable attributes and flexible attributes on the given dataset we want to classify each attribute accordingly. Now we need to what attributes are marked and unmarked.

Marked are the one where flexible/Stable attribute class fall either into any one of the decision attribute class. Unmarked are the one where there is mix of classes in the flexible/Stable attribute i.e.., some of the class values fall into one class and some fall into another class.

Certain rules are generated from the marked ones

Using unmarked ones, we generate all possible rules and their support and confidence.

Second loop:

Now combine unmarked 2 element sets and repeat the above process again. We keep on repeating the procedure until we generate all marked values.

Table 1. Sample Dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | F | G | C | A |
| x1 | f1 | g2 | c1 | a1 |
| x2 | f2 | g1 | c2 | a2 |
| x3 | f2 | g2 | c1 | a1 |
| x4 | f1 | g1 | c2 | a2 |

Let A be the decision attribute:

a1 = {x1, x3}   a2 = {x2, x4}

Let us assume G be the stable attribute

g1= {x2, x4} (marked a2) g2 = {x1, x3} (marked a1)

F, C are the flexible attributes

f1 = {x1, x4} (Unmarked) f2 = {x2, x3} (Unmarked)

c1 = {x1, x3} (Marked, a1) c2 = {x2, x4} (Marked a2)

First Loop:

Certain Rules:

g1->a2

g2->a1

c1->a1

c2->a2

Possible Rules:

f1->a1 ½ 50%

f1->a2 ½ 50%

f2->a1 ½ 50%

f2->a2 ½ 50%

Second Loop:

Since there are can’t be two sets to obtain (left with only f attribute) we stop at second loop

Action rules:

From c1->a1 and c2->a2 we can generate action rule such that

c (1->2) -> a (1->2)

Similarly using other certain rules, we can generate action rules

Action rules:

 By using all the certain rules generated from LERS Algorithm we generate action rules. An action rule is generated from two certain rules as follows:

**AR(certainRules, decisionFrom, decisionTo)**

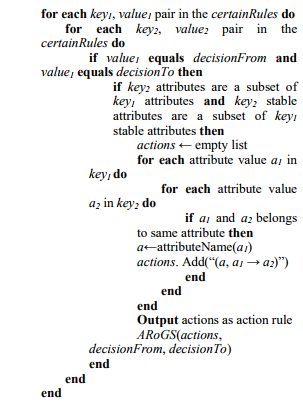


Figure 3. Action Rules Algorithm in a distributed environment using MapReduce [11]

Flexible Attribute (value1 -> value2) ^ Stable Attribute (constant value) -->

Decision Attribute(value1->value2)

In the above action rule generated to change decision attribute from value1 to value2 we can state that we need to change flexible attribute from value1 to value2 keeping stable attribute’s value constant.

Using the certain rules, we generate action rules as stated above.

Pseudo code for random forest

* Randomly select K features form total M features where K is less than M
* Among K features, calculate the node D using the best split point
* Split the node into daughter nodes using the best split
* Repeat the above steps unit L number of nodes has been reached
* Build the forest by repeating the above steps for N number of times to create N number of trees.

Pseudo code for random forest prediction

* Take the test features and use the rules of each randomly created decision tree to predict the outcome and store the predicted outcome
* Calculate the votes for each predicted outcome
* Assume the highest voted outcome as final prediction

**Reduce (Key, values)**

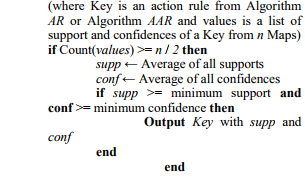


Figure 4. Random Forest Algorithm in Reduce part of MapReduce combines Action Rules from multiple mappers [11]

Action rules using MapReduce:

MapReduce framework aims to enable large scale distributed computing across multiple clusters [9]. MapReduce works by breaking the processing into two phases: the map phase and the reduce phase. Each phase has key-value pairs as input and output, the types of which may be chosen by the programmer. The programmer also specifies two functions: the map function and the reduce function [10].

Mapper: Data is split into different block, upon each block LERS algorithm is applied and action rules are generated. Those all action rules from different blocks are combined and sent to reducer function

Reducer: All the action rules generated are passed through random forest algorithm which generate the action rule set

Association Action rules using MapReduce:

**Association Action Rules Algorithm:**

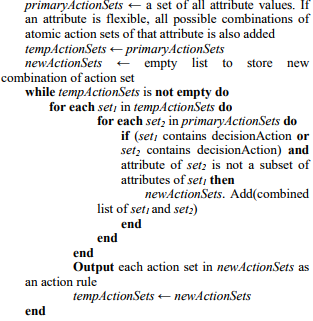


Figure 5. Association Action Rules Algorithm in a distributed environment using MapReduce [11]

Mapper: Data is split into different block, upon each block association action rule algorithm is applied and association action rules are generated. Those all action rules from different blocks are combined and sent to reducer function

Reducer: All the action rules generated are passed through random forest algorithm which generate the association action rule set

IV. EXPERIMENT AND RESULTS

Car Evaluation dataset and Mammographic-mass dataset, from the Machine Learning Repository from University of California, Irvine were used to experiment the MR-Random forest algorithm.

The dataset had the following characteristics:

The Car Evaluation dataset [12] is donated by Prof. Dr. Marko Bohanec, from Department of Knowledge Technologies, Jozef Stefan Institute, in Liublijana, Slovenia. The dataset is used to classify cars on their acceptability. It had 7 attributes in total - buying price, maintenance cost, number of doors, number of persons to carry, the size of its luggage boot, car safety and class (decision attribute). The Car Evaluation dataset has 1728 instances, and 7 attributes, as shown in Table 5. Action Rules that are extracted from this dataset are used to suggest actions to be undertaken. For example, if the car user would like to change the cars class from ‘unacc’ to ‘acc’ the changes in the attributes that are necessary for the change to take place. An example Action Rule extracted from this dataset is:

(maint, mainthigh -> maintlow) ^ (safety, safetymed -> safetymed) ^ (persons, personsmore -> persons4) ^ (lug\_boot, lug\_bootbig -> lug\_bootsmall) ==> (label, labelacc -> labelunacc) 2 [Support: 4.0, Confidence: 100.0%]

The above action rule means that: if the maintenance price of the car changes from high (mainthigh) to low (maintlow), and safety remains at medium level and the number of persons it can carry changes from personsmore (more than 4 persons) to persons4(4 persons), and the luggage boot changes from big(lug\_bootbig) to small(lug\_bootsmall), then the decision attribute (class) value is expected to change from acceptable (acc) to unacceptable (unacc). A total of 2 tuples (objects) support this rule, and we are 100% confident in the validity of this rule. In our experiment, the attributes {Safety} is designated as Stable Attribute, and the attributes {Persons, Luggage Boot} are designated as Flexible Attributes, and the attribute Class is designated as the decision attribute, which is also a flexible attribute.

Table 2. Properties and attributes for Car Evaluation Dataset

|  |  |
| --- | --- |
| **Property** | **Value** |
| Number of Instances | 1728 |
| Attribute 1 – Buying | Vhigh, high, med, low |
| Attribute 2 – Maint | Vhigh, high, med, low |
| Attribute 3 – Doors | 2, 3, 4, 5, more |
| Attribute 4 – Persons | 2, 4, more |
| Attribute 5 – lug\_boot | Small, med, big |
| Attribute 6 – Safety | Low, med, high |
| Class | Unacc, acc, good, vgood |

The Mammographic-Mass dataset [16] is donated by Matthias Elter Fraunhofer, Institute for Integrated Circuits (IIS), Image Processing and Medical Engineering Department (BMT) and Prof. Dr. Rdiger Schulz-Wendtland from the Institute of Radiology at the University Erlangen-Nuremberg, Germany. This dataset is used for Discrimination of benign and malignant mammographic masses based on BI-RADS attributes and the patient's age. The dataset had 6 attributes in total – BI RADS, Age, Shape, Margin, Density and Severity. The dataset contains a total of 961 instances out of which 516 belong to the class Benign and 445 belong to the class malignant. Action rules extracted from this dataset can be used to evaluate actions to be undertaken, to re-classify a mammographic mass lesion (tumor) from 1(malignant class) to 0 (benign class). An action rule extracted from this dataset looks like the following:

(margin, margin1 -> margin5) ^ (shape, shape3 -> shape3) ^ (density, density2 -> density3) ==> (severity, severity0 -> severity1) -> 2 [Support: 2.0, Confidence: 100.0%]

Table 2. Properties and attributes for Mammographic Masses Dataset

|  |  |
| --- | --- |
| **Property** | **Value** |
| Number of Instances | 961 |
| Attribute 1 – BI RADS assessment | 1 to 5 (ordinal, non-predictive) |
| Attribute 2 – Age | Patients age in years (Integer) |
| Attribute 3 – Shape | mass shape: round=1 oval=2 lobular=3 irregular=4 (nominal) |
| Attribute 2 – Margin | mass margin: circumscribed=1 microlobulated=2 obscured=3 ill-defined=4 spiculated=5 (nominal) |
| Attribute 2 – Density | mass density high=1 iso=2 low=3 fat-containing=4 (ordinal) |
| Attribute 2 – Severity | benign=0 or malignant=1 (binominal, goal field) |

The above rule means that: if the Margin of the lesion (tumor) changes from 1 to 5, and the shape remains shape3 (lobular) and the Density of the lesion (tumor) changes from 2(iso) to 3(low), then the severity (decision attribute) is expected to change from value 0 (benign) to value 1 (malignant). A total of 2 tuples (objects) support this rule, and we are 100% confident in the validity of this rule.

Missing attributes were present in the dataset. The details of the missing values are given in the table 3.

The results of the experiments for the car evaluation dataset and mammographic dataset are given in tables 4 and 5 respectively.

Table 3. Missing Values in Mammographic Masses Dataset

|  |  |
| --- | --- |
| **Attribute** | **Number of missing values** |
| BI RADS assessment | 2 |
| Age | 5 |
| Shape | 31 |
| Margin | 48 |
| Density | 76 |
| Severity | 0 |

For the Car evaluation dataset, ARoGS algorithm took 2.09 minutes on a single node and it took 0.99 minutes on 6 nodes. The AAR algorithm took 13.98 minutes on a single node and 4.5 minutes on 6 nodes. For the mammographic masses dataset, ARoGS algorithm took 0.73 minutes on a single node, and 0.21 minutes on 8 nodes. The AAR algorithm took 11.7 minutes on a single node, and it took 6.15 minutes on 8 nodes.

The processing times shown in tables 4 and 5 indicate that: the more the number of nodes used for parallel processing in map reduce, the faster our algorithms run in a distributed environment, compared to a single node (a single machine). We can also say that ARoGS algorithm generates the Action Rules much faster than the AAR algorithm does.

Table 4. Processing time of ARoGS and AAR algorithms for multiple nodes for Car Evaluation dataset

|  |  |  |
| --- | --- | --- |
| **Nodes** | **ARoGS (in min)** | **AAR (in min)** |
| 1 | 2.09 | 13.98 |
| 6 | 0.99 | 4.5 |

Table 5. Processing time of ARoGS and AAR algorithms for multiple nodes for Mammographic Masses dataset

|  |  |  |
| --- | --- | --- |
| **Nodes** | **ARoGS (in min)** | **AAR (in min)** |
| 1 | 0.73 | 11.7 |
| 8 | 0.21 | 6.15 |

This comparison of Action Rules produced by ARoGS and AAR is performed in Job3 of our proposed method as shown on Figure 1. Job3 produces the final list of Action Rules presented to the user.

V. CONCLUSION

The MR Random Forest Algorithm used in the experiment is a highly robust way of discovering Action Rules in a distributed environment using Hadoop map reduce framework. It significantly improves the processing time of the dataset and discovers Action Rules at a much higher pace than the traditional algorithms used for Action Rules discovery. It would be impossible to discover all the Action Rules on a single machine if the dataset is huge. Hence this MR Random Forest Algorithm is highly recommended when using large datasets for Action Rules discovery.

Action Rules are an essential part of any industry relying on data. They discover hidden rules in the dataset that cannot be detected otherwise. Datasets are getting larger and larger these days and companies are storing massive amounts of data on their servers. Hence, it is important that their action rules discovery process also scales to the data and using Action Rules in a distributed environment can be very useful if not essential.

REFERENCES

[1] Hegland, Markus. "Data mining techniques.", Acta numerica 10, pp. 313-355, 2001.

[2] Raval, Kalyani M. "Data Mining Techniques." International Journal of Advanced Research in Computer Science and Software Engineering 2, no. 10, 2012.

[3] Han, Jiawei, Jian Pei, and Micheline Kamber. “Data mining: concepts and techniques.”, Elsevier, 2011.

[4] Koh, Hian Chye, and Gerald Tan. "Data mining applications in healthcare." Journal of healthcare information management 19, no. 2, pp. 65, 2011.

[5] Blumenstock, Joshua E. "Size matters: word count as a measure of quality on wikipedia." In Proceedings of the 17th international conference on World Wide Web, pp. 1095-1096, 2008.

[6] Liaw, Andy, and Matthew Wiener. "Classification and regression by randomForest." R news 2, no. 3, pp. 18-22, 2002.

[7] Grzymala-Busse, Jerzy W., and Ming Hu. "A comparison of several approaches to missing attribute values in data mining." In International Conference on Rough Sets and Current Trends in Computing, pp. 378-385. Springer, Berlin, Heidelberg, 2000.

[8] Grzymala‐Busse, J.W. and Yao, Y., Probabilistic rule induction with the LERS data mining system. International Journal of Intelligent Systems, 26(6), pp.518-539, 2011.

[9] Wang, Lizhe, Jie Tao, Rajiv Ranjan, Holger Marten, Achim Streit, Jingying Chen, and Dan Chen. "G-Hadoop: MapReduce across distributed data centers for data-intensive computing." Future Generation Computer Systems 29, no. 3, pp. 739-750, 2013.

[10] White, Tom. Hadoop: The definitive guide. " O'Reilly Media, Inc.", 2012.

[11] Angelina A. Tzacheva, Arunkumar Bagavathi, Punniya D. Ganesan (2016) MR – Random Forest Algorithm for Distributed Action Rules Discovery unpublished.

[12] M. Lichman (2013), UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.