ISSN: 2366-1313

A STUDY FOR BIG DATA USING DISSEMINATED FUZZY DECISION TREES

[1]Dr. Prasadu Peddi, [2]Mr. Jaibir Singh

[1&2]Assistant Professor, Dep of CSE, Shri Jagdishprasad Jhabarmal Tibrewala University.

Abstract: - *We propose a definition for the Fuzzy Prolog Language that models interval-valued Fuzzy Logic and subsumes earlier approaches since it uses an accurate representation of truth value that is based on the union of intervals from real numbers. It's constructed by general operators that can be used to model various logics. We provide the procedural and declarative semantics of Fuzzy Logic programs. Additionally, we describe the design of an interpreter to this language that is based on CLP(R). We have integrated uncertainty into the Prolog program in an straightforward method thanks to the constraints system. The system is built on a syntactic extension from the original source code in this Prolog compilation.*

Keywords: *Big data, Fuzzy logic and FDT*

1.INTRODUCTION

Decision trees are widely utilized for classifiers that are used successfully in various application areas like security assessment [1, health system [2 as well as road congestion. The widespread use of these trees comes in large part because of the ease of their learning model. Furthermore, decision trees are considered to be among the best capable of being interpreted classifiers [4],[5that is, they are able to explain the way in which an output is derived from inputs. In addition, the process of tree learning typically has only a few parameters to be altered. Many algorithms include

have been suggested in recent decades to create decision trees. The majority of these are improvements or extensions of the popular ID3 suggested by Quinlan and colleagues. [6 and CART suggested by Brieman and al. [7]. A decision tree every inner (non-leaf) node signifies the testing of an attribute. Each branch is the result that was determined by the tests, every leaf (or terminal) node is tagged with an identifier for the class.

Many studies have investigated an opportunity to integrate decision trees into theories of fuzzy sets in order to tackle

ISSN: 2366-1313

uncertainty [8, 9and [9], resulting in known as fuzzy decision trees (FDTs). Contrary to Boolean choice trees, the nodes of FDTs is defined with a fuzzy set, rather than an actual set. This means that each node can trigger different branches and attain several leaves. Each of Boolean or fuzzy trees can be created using a top-down technique that divides the data used for training into homogeneous subsets. This is subsets of instances belonging to the same class. As with traditional decision trees FDTs are classified into two major groups dependent on the method of splitting employed in the generation of children nodes from the parent node [11] There are two kinds of split trees: Binary (or two-way) split trees as well as multi-way split tree. Binary split trees are distinguished by recursively splitting attributes into subspaces, so that every parent node is linked precisely to 2 child nodes. However multi-way split trees divide the space into several of subspaces, so that each parent creates generally at least two children nodes. Because a tree with multi-way splits is able to be transformed into an unidirectional tree, using a multi-way split doesn't seem to bring any advantages. It is important to remember that a binary split suggests that an attribute could be used multiple times on the same route from root to the leaf. So it is true that a binary split tree tends to be more complex and can be difficult to understand than a tree split in multiple ways. Additionally, in certain domains multi-way splits may produce more accurate trees, but since multi-way splits tends to disperse the training data quickly, they usually require greater data volumes to function effectively.

In general, FDT learning algorithms require that a fuzzy partition have been defined for every continuous attribute. Because of this, continuous attributes are typically discretized by optimizing specially defined indexes. Discretization has a significant impact on the effectiveness of classifiers and, therefore, should be implemented with diligence. The authors conducted an interesting analysis that examined how different methods of discretization affect the precision and difficulty (in terms of the number of nodes) of FDTs generated using a range of well-known fuzzy partitioning techniques and various approaches to the Boolean partition that is generated by well-known algorithms for discretization, by defining various types of functions for membership. The research results on 111 different combinations show how seven are superior to the other models in terms of precision and quantity of the nodes. FDTs have been extensively used in research

ISSN: 2366-1313

papers to classify small-scale data sets. Therefore FDT methods of learning have been concentrated on improving the accuracy of classification but have often neglected space and time requirements using a variety of tasks, such as the pruning process, genetic algorithms and the calculation of the best division of the points on each node.

Therefore, these methods do not work well when handling a large quantity of data. One possible solution to using these techniques is to pick a smaller portion of data objects using any down sampling technique.

However, these methods could overlook some important information and make FDT learning strategies specifically designed for managing the whole dataset more beneficial and efficient. In our case, this is a way of explicitly addressing Big Data. The term "Big" Data refers to a phrase that is used to describe data sets that are so vast and complex that conventional techniques for processing data are insufficient. Big Data requires specific technologies to handle unstructured or semi structured data, and to scale up with standard hardware to handle growing data volumes.

To meet these challenges, a variety of solutions have been proposed over the past few time, for example: (i) cloud computing as which is an infrastructure layer to allow big data systems , to satisfy requirements regarding efficiency, cost effectiveness and capacity to expand or scale down; (ii) distributed file systems as well as NoSQL databases, to provide persistent data storage as well as management for large datasets that do not require a scheme; (iii) MapReduce and Pregel two programming models that were developed by Google to simplify the distribution of computation across large-scale clusters of computers; (iv) cluster computing frameworks, which are powerful solutions for system-level use that include Apache Hadoop and Apache Spark for distributed processing and storage as well as system as well as failure control, as well as efficient utilization of disks and bandwidth on networks.

Most studies published in the literature on mining big data include the MapReduce model along with using the Apache Hadoop and Apache Spark cluster computing frameworks. Concerning classification problems Recent research has presented a number of distributed MapReduce versions of classic algorithms, including SVM proto reduction, kNN associative classifiers, boosting decision trees and Naive Bayes classifiers, and neural networks that study

ISSN: 2366-1313

the performance of these algorithms in terms of speed up. With the increasing of the volume and quantity of data that is big researchers are constantly examining new algorithms that take into consideration not just the accuracy of classification algorithms, but also the scaleability of the algorithms proposed, few studies have combined fuzzy set theory.

The choice trees are widely used classification tools that are utilized successfully in many areas of application such as security appraisal, wellness framework as well as street movement obstruction. The popularity of choice trees is in large part due to the ease of their learning diagrams. Furthermore, they are regarded as to be among the classifiers that are most interpretable which means that they are able to define how yields are constructed from sources of information. In the end the tree-learning process generally only requires a couple of parameters to be weighed. Many have been proposed in the last few years for the production of choice trees. most of them are extensions or enhancements of the renowned ID3 suggested by Quinlan and others. and CART suggested by Brieman and others. In a CART-based choice tree the interior of each (non-leaf) hub signifies an assessment of a quality and each branch reflects the outcome from the testing, every leaf (or terminal) hub is associated with an identifier for the class.

Some works have abused the potential of coordinating choice trees using fuzzy set theories in order to mitigate vulnerabilities, resulting in the creation of soft decision trees (FDTs). They are not at all as Boolean selection trees each hub within FDTs is depicted as an oversized set, not an actual set. The result is that every event can create a variety of branches, and even achieve diverse take-offs. They both Boolean or fluffy choices are constructed using a best-down method that divides the prepared information into homogeneous subsets. That are, subsets of instances that are in similar classes. As with traditional choice trees FDTs are arranged in two main groups, based on the method used when creating tyke hubs from an existing hub twofold (or two-way) split trees as well as multi-way splits. Zweifold splits are represented by recursively splitting the typical space into two distinct subspaces, so that each hub of a parent is linked specifically with two hubs for youngster. In addition, multi-way split trees divide the space into multiple subspaces in order to ensure that each hub's parent produces typically more than

ISSN: 2366-1313

of two hubs for toddlers. Because multi-way elements can be continuously drawn as an unidirectional tree, using multiway split is not a advantage in terms of location. One must remember the fact the fact that paired split indicates that a particular characteristic could be applied a few times in the same manner from the roots to the leaves. In this way, the paired split tree is typically more extensive it is more difficult to translate than a multi-way split. Additionally, in certain areas multi-way components are more precise trees in all instances, because multi-way parts tend to separate the information for preparation quickly They generally require a larger estimate of information with regard to the final purpose to ensure that they work effectively.

2. RELATED WORK

Numerous studies have examined the ways in which a decision tree could be efficiently constructed from vast data sets. The different methods discussed in the literature could be divided into two groups one, that is, they are defined by pre-sorting the data or using approximate representations of data, such as histograms or samples. While pre-sorting methods are more precise but they're not able to accommodate large data sets or streaming data.

One of the most well-known techniques that fall into the category of first is SLIQ which was first that was first proposed in. The SLIQ approach reduces decision tree learning time, but without sacrificing accuracy through the use of an algorithm for pre-sorting in the phase of tree growth. This technique is combined with a breadth first tree-growing method to allow classification of data that reside on disks. SLIQ also employs an algorithm for tree pruning that is based on the Minimum Description Length concept, which is low-cost and produces precise and compact trees.

Yet, SLIQ requires that some records contain data in memory for all time. Since the size of the memory-based structure of data grows proportionally to the input records, this restricts the quantity of data that can be classified by the SLIQ. SPRINT, as proposed in, eliminates the limitations on memory. Additionally, it has been developed so that it can be quickly parallelized and achieve high scaling.

In the second group that is the second category, the BOAT algorithm is an innovative method of tree construction which creates an initial tree by using a tiny portion of data, and then refines it until it

ISSN: 2366-1313

produces an end-to-end tree. The authors warrant that there will not be any differences from the actual tree (i.e. the actual tree that was constructed by

Analysing all the data the traditional manner) is identified and rectified. The different branches of the tree constructed within two scans over the training data. In a decision tree building algorithm, a technique known as SPIES is suggested. SPIES restricts the possible split-points by using a small sample from an array of values, dividing the data into intervals, and calculates class histograms of the possible split points. This decreases the complexity of space of the algorithm, as well as the communication costs between processors.

The many methods to parallelize the process of learning by decision trees can be classified into four main categories which are: 1.) horizontal or data-based parallelism divides the data in such a way that different processors are able to work with various examples while the second) vertical or feature-based parallelism allows different processors to take into consideration different aspects and Iii) tree, or task node-based, parallelism allocates the tree nodes among the slave processors , and the fourth) hybrid parallelism blends horizontal and vertical parallelism in the initial phases of tree building and task parallelism toward the close. The authors explain the possibility of parallelizing two distinct algorithms for learning decision trees such as C4.5 and the linear discriminant univariate tree suggested using vertical, horizontal as well as the task of parallelisation. The results of experiments show that the performance of the parallelization is highly dependent on the data, however the node-based approach generally provides significant speed improvements. The authors use an approximate representation, as well as horizontal parallelism. The heart part of this algorithm is an online procedure to build histograms using streams of data processed by processors. Histograms represent compressed versions of data that can be sent into a master processor that has little or no communications complexity. The master processor incorporates the information from all processors and decides the terminal nodes that need to be split and in what order. A brief overview of the algorithms for learning decision trees that are proposed to deal with large-scale datasets is presented in. Based on previous work of distributed decision trees the FDT learning algorithms presented in this paper

ISSN: 2366-1313

take advantage of both task and horizontal parallelism.

In recent several years, a variety of decision tree-based learning algorithms were proposed to handle massive amounts of data. They have adopted the MapReduce approach on foundation of Apache Hadoop. MapReduce is based on functional programming. It divides the computational process into two major stages, specifically Map and Reduce that communicate using pairs of hkey and valuei. MapReduce is a MapReduce implementation of a distributed decision tree suggested uses, for example four map-reduce steps. The initial stage is a scan of the data for the data structures that will be used in the remaining three stages. The stages are repeated for (i) choosing the best quality attribute (ii) making changes to the stats of the new nodes, and (iii) expanding the tree. The results of the experiments discussed within the article are restricted to scalability analysis by altering the number of nodes as well as the size of the data (up to 3 million instances). The efficiency of the decision trees in managing large data has been proven in real-world applications like the prediction of stock futures and clinical decision assistance. Others use decision trees to create ensembles of classifiers, such as random forest.

3. METHODOLOGY

In this section, we first introduce the FDT and the necessary notations used in the paper and then we describe both the MapReduce programming model and the Apache Spark cluster computing framework. This framework is exploited in our DFDT.

A. Fuzzy Decision Tree Instance classification consists of assigning a class Cm from a predefined set C = fC1;CMg of M classes to an unlabeled instance. Each instance can be described by both numerical and categorical attributes. Let X = fX1; ;XF g be the set of attributes. In case of numerical attributes, Xf is defined on a universe Uf <. In case of categorical attributes, Xf is defined on a set Lf = fLf;1;Lf;Tf g of categorical values. An FDT is a directed acyclic graph, where each internal (non-leaf) node denotes a test on an attribute, each branch represents the outcome of the test, and each leaf (or terminal) node holds one or more class labels. The topmost node is the root node. In general, each leaf node is labeled with one or more classes Cm with an associated weight wm: weight wm determines the strength of class Cm in the leaf node.

ISSN: 2366-1313

Let TR = f(x1; y1); (x2; y2); :::; (xN; yN)g be the training set, where, for each instance (xi; yi), with i = 1; :::;N, yi 2 C and xi;f 2 Uf in case of continuous attribute and xi;f 2 Lf in case of categorical attribute, with f = 1; :::; F. FDTs are generated in a top-down way by performing recursive partitions of the attribute space.

Algorithm 1 shows the scheme of a generic FDT learning process. The Select Attribute procedure selects the attribute used in the decision node and determines the splits generated from the values of this attribute. The selection of the attribute is carried out by using appropriate metrics, which measure the difference between the levels of homogeneity of the class labels in the parent node and in the child nodes generated by the splits. The commonly used metrics are the fuzzy information gain, fuzzy Gini index, minimal ambiguity of a possibility distribution, maximum classification importance of attribute contributing to its consequent and normalized fuzzy Kolmogorov-Smirnov discrimination quality measure. In this paper, we adopt the fuzzy information again, which will be defined in Section IV-B. The splitting method adopted in the Select Attribute procedure determines the attribute to be selected and also the number of child nodes.

In the literature, both multi-way and binary splits are used. We have implemented both the approaches and evaluated their pros and cons. Once the tree has been generated, a given unlabeled instancebx is assigned to a class Cm 2 C by following the activation of nodes from the root to one or more leaves. In classical decision trees, each node represents a crisp set and each leaf is labelled with a unique class label. It follows that bx activates a unique path and is assigned to a unique class. In FDT, each node represents a fuzzy subset. Thus, bx can activate multiple paths in the tree, reaching more than one leaf with different strengths of activation, named matching degrees. Given a current node CN, the matching degree mdCN(bx) of bx with CN is calculated as:

$$md^{CN}(\hat{\mathbf{x}}) = TN(\mu^{CN}(\hat{x}_f), md^{PN}(\hat{\mathbf{x}})) \qquad (1)$$

where TN is a T-norm, CN(bxf) is the membership degree of bxf to the current node CN, which onsiders Xf as splitting attribute, and mdPN(bx) is the matching degree of bx with the parent node PN.

The association degree ADLN m (bx) of bx with the class Cm at leaf node LN is calculated as:

ISSN: 2366-1313

where mdLN(bx) is the matching degree of bx with LN and wLN m is the class weight associated with Cm at leaf node LN. In the literature, different definitions have been proposed for weight wLN m. Further, it has been proved that the use of class weights can increase the performance of fuzzy weighed vote: the class corresponds to the maximum total strength of vote. The total strength of vote for each class is computed by summing all the activation degrees in each leaf for the class. If no leaf has been reached, the instance bx is classified as unknown.In 2004, Google proposed the MapReduce programming framework [5] for distributing the computation flow across large-scale clusters of machines, taking care of communication, network bandwidth, disk usage and possible failures.

At high level, the framework, which is based on functional programming, divides the computational flow into two main phases, namely Map and Reduce, organized around hkey; valuei pairs. When the MapReduce execution environment runs a user program, the framework automatically partitions the data into a set of independent chunks, that can be processed in parallel by different machines. Each machine can host several Map and Reduce tasks. In the Map phase, each Map task is fed by one chunk of data classifiers. To determine the output class label of the unlabeled instance bx, two different approaches are often adopted in the literature: maximum matching: the class corresponds to the maximum association degree calculated for the instance.

and, for each hkey; valuei pair as input, it generates a list of intermediate hkey; valuei pairs as output.

In the Reduce phase, all the intermediate results are grouped together according to a key-partitioning scheme, so that each Reduce task processes a list of values associated with a specific key as input for generating a new list of values as output. In general, developers are able to implement parallel algorithms that can be executed across the cluster by simply defining Map and Reduce functions.

4. THE PROPOSED DISTRIBUTED FUZZY DECISION TREE FOR BIG DATA

The distributed method proposed lets you manage a vast amount of data. Performing the splitting of a set of nodes dramatically reduces the number of scans across the training set, however it also requires a higher amount of memory, and a greater processing time per iteration (the cost of computation is minimized by aggregating

ISSN: 2366-1313

and collecting the essential data). So, the total number of nodes that can be processed in parallel on every iteration, is determined by the availability of memory for the cluster. Naturally, the greater number of categorical values and fuzzy sets that are defined by fuzzy partitioning the greater the amount of memory utilized to store the statistics and the lesser amount of the nodes which can be processed in parallel on each repetition.

with a complete dataset, varying the number of CUs; iii) ability to efficiently accommodate an increasing dataset size.

As shown in Table I, we employed 10 well-known big datasets freely available from the UCI2 repository. The datasets are characterized by different numbers of input/output instances (from 1 million to 11 million), classes (from 2 to 50), and attributes (from 10 to 41). For each dataset, we also reportthe number of numeric (num) and categorical (cat) attributes.

5. CONCLUSION

We have developed an dispersed fuzzy decision tree (FDT) learning algorithm that is designed in accordance with the MapReduce programming model that generates the binary (FBDT) as well as multi-way (FMDT) FDTs using large data. We first proposed an innovative distributed fuzzy discretise that creates strong fuzzy partitions for every continuous attribute, based on fuzzy information and entropy. We then have presented a distributed implementation the FDT learning algorithm that utilizes fuzzy information gain to select the attributes that will be used to determine the nodes. The algorithm has been implemented by us FDT learning algorithm on Apache Spark. Apache Spark framework.

Experiments conducted on 10 large datasets from the real world demonstrate that our method can be used to attain speedup and scalability levels that are like the ideal figures. It is important to note that these results can be achieved without the need for specific hardware, rather using a small number of personal computers that are connected via Gigabit Ethernet. The results were contrasted with those obtained using the distributed decision tree (DDT) which is a part of the MLlib library that is part of Apache Spark framework and by Chi-FRBCS BigData, which is a MapReduce distributed fuzzy rule-based classification system. For comparison, we've examined accuracy, complexity, as well as the time to execute. We have discovered that FBDT

ISSN: 2366-1313

outperforms Chi-FRBCS BigData, FMDT and DDT regarding accuracy. For complexity FBDT as well as DDT employ smaller (generally an order of the magnitude) number of nodes than FMDT. Furthermore, the number rules that are extracted using three trees of decision, are typically smaller than that of Chi-FRBCS-BigData. From a time perspective both FBDT and FMDT have similar computation times, however both are less efficient than DDT (not unexpected, considering fact that both FBDT and FMDT use an unreliable partitioning process and handle more data because of fuzzy logic) However, they're more efficient than Chi-FRBCS BigData. The time for computation also grows roughly linearly with the amount of computation units and instances.

In the paper's overall argument, the primary reason for the idea of FBDT as well as FMDT is to create effective and efficient classifiers for managing large data. The distribution of data across computer cluster results in parallelization of fuzzy decision tree learning process and, consequently, the ability to generate trees faster. Therefore, FBDT and FMDT have a place in every domain where decision trees must be created quickly using a lot of data. For instance, increasing number of sensors being used and the subsequent need to gain useful information from information gathered from these sensors has led to an increase in demand for data mining techniques that stream data, and possibly that are able to control concept drift. Most approaches used in this case employ sliding windows that are size that is variable or fixed and a retraining mode. A window is kept that includes the most recent instances, and older examples are eliminated in accordance with a set of guidelines. The retraining method discards the model and creates a new model from scratch using buffered data from windows. A comprehensive overview of the analysis of streaming data and concept drift adaptive is available. Our fuzzy decision tree-based learning algorithms are especially suitable for the retraining mode of learning particularly for windows that are large.

We conclude that the research described in this paper is the very first comprehensive study of applying FDTs to big data looking at the binary and multi-way splits. We anticipate that the results from the experiments will serve as a reference to further research into this area.

ISSN: 2366-1313

REFERENCES

[1] R. Diao, K. Sun, V. Vittal, R. J. O'Keefe, M. R. Richardson, N. Bhatt, D. Stradford, and S. K. Sarawgi, "Decision tree-based online voltage security assessment using PMU measurements," IEEE Transactions onPower Systems, vol. 24, no. 2, pp. 832–839, 2009.

[2] Prasadu Peddi (2018), Data sharing Privacy in Mobile cloud using AES, ISSN 2319-1953, volume 7, issue 4.

[3] Y. Zheng, L. Liu, L. Wang, and X. Xie, "Learning transportation mode from raw gps data for geographic applications on the web," in Proceedings of the 17th international conference on World Wide Web, 2008, pp. 247–256.

[4] J. Han, M. Kamber, and J. Pei, Data mining: Concepts and techniques. Elsevier, 2011.

[5] L. Rokach and O. Maimon, Data mining with decision trees: Theory and applications. World scientific, 2014.

[6] J. R. Quinlan, "Induction of decision trees," Machine learning, vol. 1, no. 1, pp. 81–106, 1986.

[7] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, Classification and regression trees. CRC press, 1984.

[8] Prasadu Peddi (2017) "Design of Simulators for Job Group Resource Allocation Scheduling In Grid and Cloud Computing Environments", ISSN: 2319- 8753 volume 6 issue 8 pp: 17805-17811.

[9] Prasadu Peddi (2016), Comparative study on cloud optimized resource and prediction using machine learning algorithm, ISSN: 2455-6300, volume 1, issue 3, pp: 88-94.

[10] J. R. Quinlan, C4.5: Programs for Machine Learning. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[11] X. Liu, X. Feng, and W. Pedrycz, "Extraction of fuzzy rules from fuzzy decision trees: An axiomatic fuzzy sets (AFS) approach," Data & Knowledge Engineering, vol. 84, pp. 1–25, 2013.

[12] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: Data mining, inference and prediction," The Mathematical Intelligencer, vol. 27, no. 2, pp. 83–85, 2005.