MÜHENDİSLİK MİMARLIK FAKÜLTESİ

Uluslararası Mühendislik Araştırma ve Geliştirme Dergisi
International Journal of Engineering Research and Development

UMAGD, (2021) 13(1), 140-151.

10.29137/umagd.763490

Cilt/Volume:13 Sayı/Issue:1 Ocak/January 2021

Derleme Makale / Review Article

Decision Trees in Large Data Sets

Zeynep Çetinkaya[1], **Fahrettin Horasan**[1*]

[1]*Engineering Faculty, Computer Engineering Department, Kırıkkale University, Kırıkkale, Turkey*

Başvuru/Received: *03/07/2020* ***Kabul / Accepted***: *17/12/2020* ***Çevrimiçi Basım / Published Online:*** *18/01/2020*

Son Versiyon/Final Version: *18/01/2020*

Abstract

Data mining is the process of obtaining information, which is used to identify and define the relationships between data of different qualities. One of the important problems encountered in this process is the classification process in large data sets. Extensive research has been done to find solutions to this classification problem and different solution methods have been introduced. Some decision tree algorithms are among the structures that can be used effectively in this field. In this article, various decision tree structures and algorithms used for classification process in large data sets are discussed. Along with the definitions of the algorithms, the similarities and existing differences between them were determined, their advantages and disadvantages were investigated.

Key Words

"Decision trees, Decision tree algorithms, Big data sets, Scalable decision trees"

***Sorumlu Yazar:** *fhorasan@kku.edu.tr*

1. Introduction

Data mining is the process of obtaining unknown and valuable information based on existing data. It can also be defined as information discovery in databases (Aytekin et al., 2018). Large amounts of data and important information to be obtained from this large-scale data play an important role in the emergence of data mining (Han and Kamber, 2006). Data mining includes many methods and algorithms for information discovery. All of the transactions for the determination of relationships, changes and rules that are hidden in big data and difficult to detect are referred to as the data mining process. The most important goal of this process is to reach these logical rules in the fastest and most accurate way (Aytekin et al., 2018).

Data Warehouses are used for data mining techniques to work effectively and to give fast and accurate results. Data warehouses can be considered as well-defined databases. New databases created to minimize the problems that may arise in the processing of data in normal databases are called data warehouses. Only data to be used in the analysis are transferred to this new structure (Aytekin et al., 2018). It is an effective method that prevents data that can prevent analysis or cause false results by cleaning data inconsistencies, dirty data and noise. Data mining can also be defined as finding meaningful patterns and rules by analyzing large amounts of data on these data warehouses (Berry and Linoff, 1997)

Data mining includes lots of technic which are under development like statistics, database technology, neural networks, decision trees, artificial intelligence, machine learning, genetic algorithms and visual techniques. In addition, data mining is applied in many areas such as marketing, finance, banking, customer relations, production and health (Han and Kamber, 2006; Chein and Chen, 2008). The most common data mining techniques include classification and regression. In these two data analysis methods, grouping is made according to the logical rules sought in large data sets by using pre-classified samples. While the classification process is used to estimate categorical values, the regression process is the methods used in the estimation of continuous values. Decision trees are one of the main techniques used in these methods, which play a role in creating models that reveal important classes and predict future data trends. (Aytekin et al, 2018).

2. Decision Trees

Decision trees are a frequently used classification and regression technique because of their easy interpretation compared to other classification methods, their realization at lower costs, the ease of integration with databases and a good level of reliability (Chein and Chen, 2008). In addition, leaves shown as decision rules in decision trees can be easily interpreted by people working in this field and this method is used effectively in high dimensional data. (Chaudhuri et al., 1997).

In decision trees, classification processing is carried out in two steps: learning and classification. In the learning step, the so-called "training data set" is used, in which the results matched to the values are known. This training data is transmitted to decision tree classification algorithm and analyzed by algorithm for create the model. These models learned as a result of the analysis are defined as classification rules or decision trees. With the end of the learning process, the second step, the classification step, is started. In the classification step, the data set called "test data set" is used. This step is performed in order to determine the correctness of the created classification rules or decision tree. If accuracy is acceptable, rules or a decision tree can be used to classify new data (Özekeş and Çamurcu, 2002; Çalış et al., 2014).

A decision tree has a real tree-like structure, starting with a variable called "root node", established in a hierarchical relationship. This root node can be divided into two or more branches within the framework of certain rules. With each branch created, certain classes are represented throughout a class or node that leaves the root node. In each branching-dividing process, a question is asked to the system and an answer is received regarding the range or class of the variable. Depending on the answer to this question, branching-dividing is carried out and new sub-nodes are created. After each new node created, a branching operation is performed again using the class or range of another variable. In each transaction, the split node is called "main node", and the nodes formed as a result of the split are called "lower node / leaf". This branching process continues until the classification ends or if it encounters a cutting rule (Nispet et al., 2009).

The most important point in the creation of a decision tree is to determine which criteria or attribute values will use for the branching in the tree will be performed. There are many approaches in the literature that can be used to make the right choice at this point. Some of the approaches used in determining the branching criteria; Gini index is seen as Towing rule (Swain and Hauska, 1977), Chi-Square probability table statistics and information gain & information gain rate.

Among the branching criteria, knowledge gain and knowledge gain rate method find more usage in studies than other methods. In this method, information theory consisting of entropy rules is used in order to determine which feature of the branching tree will be made. “Entropy” can be defined as the measure of irregularity or uncertainty of a system (Kavzoğlu and Çölkesen, 2010).

The concept of entropy is a concept used in many fields of science, the person who adapts this concept to information theory is known as Claude Shannon. (Shannon,1948) According to Shannon, if the probability of occurrence of A event is P (A), the entropy value is expressed as -log x P(A) (Kacar et al., 2013,,Varçın er al., 2016). The more the value obtained using the entropy formula, the more uncertain and unstable the results are considered(). For this reason, it is desired that Entropy measure is at the minimum level in decision trees. Entropy is expressed as equation1:

$$Entropy = -\sum_{i=1}^{n} P_i \, log_2(P_i) \qquad (1)$$

Another important issue in the formation of decision trees is as much as the realization of the branching process. After the classification process, the structure of the decision tree is formed by organizing the training version to form clusters containing a single subclass. Therefore, a very large and complex tree structure emerges. In such cases, it may be possible to drop a subtree in the decision tree structure and replace it with a leaf. The process performed in these and similar ways is called "Pruning the Decision Tree" (Mingers, 1989).

Pruning is the removal the parts that do not affect the accuracy of the classification process from a decision tree. With this process, a simpler and easier to understand decision tree is obtained by reducing the complexities that may occur in the decision tree. Two methods are generally used to simplify the decision tree structure through pruning. The first of these two methods is called the "pre-pruning" method. In the pre-pruning method, pruning is performed simultaneously, while the tree structure is formed. The other method is the method called "last pruning" where pruning is done after the tree structure is created (Kavzoğlu and Çölkesen, 2010; Breiman et al., 1984).

“Decision Tree Algorithms” are separated from each other thanks to different ways to select root, knot and branching criteria (Bounsaythip and Rinta, 2001). Algorithms, the foundations of which are laid by the AID method, have been developed over time and various algorithms have been proposed to be used in decision trees. Some of these developed algorithms can be listed as AID, CHAID, CART, ID3, C4.5 and C5.0 (Akpınar, 2000).

2.1. AID Algorithm

Known as the first algorithm and first software created based on the decision tree structure is the AID (Automatic Interaction Detector) algorithm, was proposed in 1963 by experts named Morgan and Sonquist. AID is a decision tree method based on performing the best estimation in a classification process and finding-separating those with the strongest relationship between the independent variables. In the AID algorithm, the variables in the system are divided into two according to their value, forming the tree structure. The divisions that occur by dividing into two continue repeatedly until all variables are terminated. The independent variable type can be classifier and sorter. The difference between AID and regression analysis is that it does not provide reliable information about its significance among variables (Akpınar, 2000).

2.2. CHAID Algorithm

Another decision tree algorithm is the CHAID (ChiSquared Automatic Interaction Detector) algorithm. The CHAID algorithm, which has the purpose of classification and regression and also works on statistics, was developed by G. V. Kass in 1980 (Pehlivan, 2006, s. 11). CHAID algorithm is a technique that uses chi-square test instead of entropy or gini techniques used in other algorithms to select the optimum sections to be performed in tree formation. It is a frequently preferred algorithm today due to its ability to work on categorical and continuous variables, to divide the main mass which is intact as a whole with a strong iteration algorithm into stable sub-nodes, to divide each node in the tree into two or more than two subgroups, according to the structure of the data (Aytekin et al.,2018; Oğuzlar, 2014). The fact that the CHAID algorithm structure is out of certain assumptions allows this algorithm to be used as an alternative tree diagram to binary and multinominal logistic regression models (Oğuzlar, 2004; Akpınar, 2000).

Table 1. Decision Tree Algorithms

Algorithm	Advantages	Disadvantages
CHAID	- Performs classification using chi-square tests. - It forms tree structure with multiple branching. - For preliminary, pre-pruning process with chi-square test. - The number of branches varies according to the number of categories of the predictor.	- It does not give acceptable results in continuous variables.
CART	-Gini index and Twoing criteria use for classify. - Pruning trees using a complex model predicted by cross-validation of parameters. - When the tested property has an unknown value, it looks for alternative values close to the results. -It can work with continuous target variables.	-The data needs to be prepared in advance. -Allows binary classification operations only. -The success of the operation of the operation depends on the size of the tree's complexity.
ID3	-It has a very simple structure. -It is a simple process, it can be easily applied. -Work time increases only linearly with the complexity of the problem.	-When working with continuous data and missing data, it does not give an acceptable result. -It requires more memory. -There are long call times. -There may be excessive learning (overfitting) or excessive classification.
C 4.5	- Faster than ID3 - Memory and computing efficiency are higher. - Information is used for classification and creates trees with multiple branches emerging from each node. -It can work with missing or continuous data. - Can use features with different weights. - Avoids overfitting the data. - It allows to determine how much the decision tree will be grown.	-The formation of empty branches is observed. -Branches that are considered to be unimportant occur. -It is sensitive to noise.
C 5.0	-It works faster than C4.5. - Provides less memory usage than C4.5 when creating the rule sequence. -The created rule sets are easier to understand. -Uses information-based criteria for classification. - Gives information about noise and missing data. - Can make feature selection, estimate which qualities are relevant and which are not important in classification. - It provides cross validation and reduced error rate pruning opportunities. -Supports the improvement of the trees and provides more accuracy rates. -It brought solutions to the problem of overfitting.	-For applications with very high numbers of situations, it may crash with a message such as segmentation error. -Using case weight does not guarantee that situations with higher weight of the classifier will be more accurate.

2.3. CART Algorithm

The CART (Classification and Regression Trees) algorithm was introduced into the literature of decision tree algorithms in a study by Breiman et al in 1984. In the CART algorithm, at each stage, the related cluster is separated into two subsets which is more homogeneous than this cluster. In this separation process, while gini and twoing are used for categorical variables, calculations are made according to the least squares deviation index for continuous variables (Aytekin et al., 2018; Oğuzlar, 2004). In the CART algorithm, which can work with both categorical and continuous data types, a representative variable can also be assigned for missing values (Demirel and Giray, 2019).

The CART algorithm, which can be considered as an automated machine learning method, can work with relatively few inputs in complex structures. If there is a need pruning for a complex tree structure that occurs after splitting, pruning is started with the end of the splitting and is carried out from the tip to the root. In order to obtain the most successful decision tree in this algorithm, after each pruning process, the decision tree is evaluated with a randomly selected test data and the optimum tree structure is tried to be determined (Sezer et al., 2010; Oğuzlar, 2004).

2.4. ID3 Algorithm

The ID3 (Iterative DiChaudomiser3) algorithm was developed by J. Ross Quinlan (1979). This Decision tree algorithm is basically based on the conceptual learning system (CLS) algorithm. In cases where the training set contains too many records and there are many qualities, ID3 algorithm can be used in cases where a reasonable decision tree structure can be created with few calculations. ID3 is an algorithm with a recursive structure, it cannot operate on numerical attributes or in cases where there is missing data, nor does it perform any pruning while the tree structure is created or after it is created. Basically, this algorithm is used to classify categorical attributes into a tree structure (Aytekin et al., 2018; Hssına et al., 2014).

2.5. C 4.5 Algorithm

With the study published by J. Ross Quinlan in 1993, the C4.5 decision tree algorithm, which is accepted as an advanced version of the ID3 algorithm and set out to eliminate the missing sides of the ID3 algorithm, was created. In this algorithm, created to address the missing aspects of the ID3 algorithm, data sets containing missing data can be processed and pruning can be done in complex decision trees. The C4.5 algorithm can be used for attributes with continuous values. In addition, the C4.5 algorithm contains a 'gain rate' value different from ID3 and creates a classification decision tree by recursively dividing the data into subsets (Aytekin et al., 2018; Hssına et al., 2014; Yang and Chen, 2016).

2.6. C 5.0 Algorithm

The C5.0 algorithm was obtained by Quinlan by developing the C4.5 algorithm. The advantages of this algorithm, which can be performed using categorical variables, compared to the C4.5 algorithm; faster, less memory, more precise rules, the ability to give weighs for variables and misclassification types, and to exclude variables that do not contribute to the tree's formation. It can also use memory more efficiently and does not require less learning time (Zhixian et al., 2009; Patil et al., 2012)

3. Decision Trees In Large Data Sets

Although the classification has been extensively studied in the past, the various techniques proposed for classification have not been well scaled for large data sets (Metha et al., 1996). However, the use of large data sets is important for increasing classification accuracy (Gehrke et al., 1998). Many of the existing algorithms have the limitation of training data fitting into memory. In the literature, there are algorithms developed to remove this constraint and creating more effective decision trees for classify. The first of these is the SLIQ algorithm and is specifically designed for scalability. Over time, new algorithms have been developed with the creation of the number of records that even the SLIQ algorithm can't handle. SPRINT has emerged as a method with effective parallelization that requires very little addition to the serial algorithm (Shafer et al., 1996). New algorithms have been developed such as ScalParC, RainForest, CLOUDS, BOAT, SPIES, etc. for complement and develop the missing aspects noticed in the SPRINT algorithm. Some of these algorithms will detailed mentioned in this section.

3.1. SLIQ Algorithm

The SLIQ algorithm, developed by Mehta, Agrawal and Rissanen researchers in 1996, can be used for classification in data sets with numerical and catagorical properties. This algorithm, supported by IBM Quest, divides the data set into segments using the pre-sorting technique when creating a decision tree in large data sets. In this algorithm, the sorting procedure is integrated with a broad tree growing strategy to ensure the classification of data sets resident on the disk. SLIQ also uses a cheaper and more compact tree pruning algorithm than other algorithms to achieve simplicity in complex tree structures. As a result of the combination of these techniques, SLIQ allows the data to scale and classify large data sets regardless of class, quality or number of samples.

One of the important advantage of the SLIQ algorithm is that it greatly reduces sequencing costs on nodes. SLIQ maintains a discrete order list called "class list" for each node, and each element in this list corresponds to a property in the data. Each created list element has a class tag. Since the SLIQ algorithm uses the width priority path when creating the decision tree, it creates multiple leaf-nodes simultaneously. While creating these nodes, it scans the class list that is properly sorted for each attribute and calculates the entropy value defined as the measure of irregularity or uncertainty for each value. After the entropy values are calculated, the best quality is selected for partition the data, and the decision tree structure is established by creating nodes. This process continues iteratively until the data is completely categorized.

The outstanding feature of the SLIQ algorithm compared to other classifiers is that it is fast and produces decision trees that give good results in short periods of time. For SLIQ, it can be said that it provides good scalability and performs well in large datasets containing a large number of samples and attributes (Mehta et al., 1996; Shafer et al., 1996).

3.2. Sprint Algorithm

Developed in 1996 by Shafer, Agrawal and Mehta, the SPRINT algorithm can work with very large data sets. The main goal of this algorithm is to avoid memory constraints by eliminating the relationships between the main memory size and the dimensions of the training data set. This algorithm, designed to easily parallelize while removing memory constraints, also features a fast and scalable algorithm. The SPRINT algorithm allows many different processors to work together on the same model to create a single consistent model.

SPRINT initially creates an “attribute list” consisting of an quality value, a class tag, and an array containing the record from which this value is obtained, for each property in the dataset. In large datasets if all of the data is not in memory, the generated attribute lists are kept on disk. The first list created using the training set appears as an attribute list of the root node of the classification decision tree. As the decision tree grows in line with the data, as the existing nodes are separated to create new sub-nodes, the attribute created for each node is also segmented and associated with the new sub-nodes that are created. When a created list of attributes is partitioned, the order of records in the new list created is always maintained, so the partitioned lists never require a reference again. (Shafer et al., 1996).

3.3. Sprint vs SLIQ

In order to work on large data sets, the technique of creating separate property lists using data is the first system proposed by the SLIQ algorithmIn the SLIQ algorithm, each entry in the generated list of features consists of an attribute value and an array; class labels are kept in a separate data structure, also referred to as the "class list" indexed with the created directory. Each entry in the class list has a pointer in addition to the class tag. The SLIQ algorithm does not have to rewrite these lists when splitting nodes while creating the decision tree, but if the records are reassigned to new nodes, the changing the pointer of the class list becomes necessary. Class lists are randomly created, if they are accessed frequently and updates are made, they must remain permanently in the memory for the entire processing time, resulting in serious performance drops. Also, the size of this class list grows in direct proportion to the size of the training set used, and given the situation, there will be limits on the size of the training set that can operate with the SLIQ algorithm.

The SPRINT algorithm uses different data structures than the SLIQ algorithm. The purpose of the SPRINT algorithm is to do correct classifying and effectively develop the classify operation, in datasets that are too large and difficult to process for any other algorithm. Data placement and workload balancing problems that should be taken into consideration in the classification process with any parallel algorithm can be easily solved using SPRINT. SPRINT is specially designed to eliminate dependency on data structures that are central or resident in memory; SPRINT algorithm is parallel naturally and efficiently to these design goals (Mehta et al., 1996; Shafer et al., 1996).

3.4. ScalParC Algorithm

The ScalParC algorithm is emerging as a new parallel formulation in classification methods based on decision tree structure. This algorithm, which is suitable for processing large data sets, was developed by Joshi, Karypis and Kumar in 1998, and this scalable algorithm has been shown to be scalable both in memory requirements and at runtime. The main data structures used in the ScalParC algorithm are to be listed briefly, distributed attribute lists are node tables and counting matrices.

ScalParC shows a structure that is carried out in stages, with 4 basic steps. The first of these stages is called "Find-Split-I" and at this stage the local count matrix is calculated for all continuous attribute value. To find the local counting matrix, a parallel prefix is applied in the Find-Split-I phaseIt is designed to systematize the calculation of general counting matrices for all nodes created using a processor parallel reduction process for a categorical attribute. The second stage is called “Find-Split-II”, in this phase it is decided whether a node needs further division using termination criteria. For nodes that need division, the optimum gini index is calculated. In continuous attributes, local list records are scanned to find the most suitable spread point. For a categorical attribute, the gini array is calculated by the processor, which was specified by the global count matrix in the previous stage. In order to find the best division criteria for each node, it uses a reduction operation that is parallel with this algorithm and has the best division state.

In the third phase, called the "Perform-Split-I" phase, the lists of attributes for the partitions are split, mixed buffers are created, and the distributed table of node is updated in accordance with the innovations that have occurred. Maintaining memory scalability is an important criterion at this stage. To meet this criterion, updating the node table can be performed in multiple steps The last stage is called "Perform-Split-II" and at this stage, the lists of all the attributes that do not go through the split are divided by taking one attribute at all time. For each attribute value, the generated node table is queried using the query process. Node information collected after queries is added to the table for use in dividing the next attribute (Joshi et al., 1998).

3.5. Sprint vs ScalParC

Thanks to the design of the SPRINT algorithm, it allows effective parallelization of the splitting process when creating the decision tree. Also, with SPRINT, it is enough to list the qualities once. However, due to the parallel formulation recommended for the splitting stage, a clear scaling cannot be performed in both memory requirements and runtime. In the SPRINT algorithm, registration node mapping is collected from each processor and the required attribute table is created in all processorsAt this stage, the communication load per processor is calculated as O (N). The serial working time of a classifier is also calculated as O (N). Therefore, the SPRINT algorithm cannot scale at run time. As the memory requirement per processor is found as O (N) in the studies performed, it is considered as non-scalable in memory requirements. When the high levels of the decision tree obtained in the classification are reached, the size of the hash table is in line with the size of the training data set, and this table is found in each processor.

ScalParC has a scalable algorithm design in both runtime and memory requirements, with this advantage it is called "Scalable parallel classification". Like the SPRINT algorithm, ScalParC continuous attributes lists only once and uses a similar attribute list. The main difference between the two algorithms is that ScalParC uses a distributed attribute table to implement the division step. It enables the creation of a new parallel table with the communication system used to create this table and to access this table when necessary in the transactions. Studies in this case have shown that the total communication load in the algorithm does not exceed O (N) and that the memory per processor required to apply this algorithm does not overrun O (N / p). As a result of these studies, the scalability of the ScalParC algorithm has been proven both at runtime and in memory requirements. (Shafer et al., 1996; Joshi et al., 1998).

3.6. Rainforest Algorithm

The RainForest decision tree algorithm, created by Gehrke, Ramakrishnan and Ganti in 1998, is a unifying classifier framework that contains the scalability and central properties that determine the quality value of the tree aspects of other algorithms. Concretization of this genetic algorithm with certain algorithms in the literature (C4.5, CART, CHAID, ID3, SLIQ, SPRINT and QUEST) is easily realized. According to the SPRINT algorithm, which is accepted as a fast classification algorithm in the literature, the RainForest algorithm offers a performance increase of more than five times. However, the RainForest algorithm requires that a minimum main memory be initially maintained to maintain the necessary operations in proportion to the existing variable values. Given the main memory costs available for algorithms, this requirement is at a level that can easily be met in most, but not in all classification processes.

RainForest offers an important concept, referred to as the AVC set, which stands for Attribute-Value Class. The AVC set of the attribute of a particular n node is characterized by the number of data with different attribute values of a and different classes. The AVC group of a n node is a combination set consisting of all AVC sets that have a n node. With the definition of AVC sets, the RainForest algorithm manages to distinguish scalability problems in decision trees from the quality problems of the decision tree. In providing this situation, RainForest algorithm can reveal important observations with AVC set and AVC group. The first of these observations can be explained as follows, only with the AVC group of a particular node, for most decision tree creation algorithms, there is enough information to decide whether the node is a leaf or a leafless decision node. Another observation is the determination of which tests should be used in this node. In the light of these observations, the RainForest algorithm has enabled a number of new algorithms to occur depending on how many AVC groups can be in memory.

The first algorithm proposed by RainForest emerges as the RF-Write algorithm. RF-Write algorithm assumes that existing AVC group data of the root node is stored on disks. In this algorithm, where the sub-nodes are created by applying the split selection method, The RF-Write algorithm reads all data in the database twice and works by writing all database data once for each level in the decision tree. This situation causes excessive time cost for the algorithm.

The second algorithm created by RainForest is called the RF-Reading algorithm. İn this algorithm the basic logic is not to build the decision tree using a top-down approach, iteratively. The RF-Reading algorithm creates the tree iteratively by calculating the AVC groups belonging to all of the existing nodes that will be evaluated at the same level in the decision tree at the same time. However, the use of this method may cause the problem that AVC groups do not fit in memory with all nodes of the same level of the decision tree, this situation is depending on the size of AVC groups. As a result of the studies carried out as a solution to this problem, it was proposed to use multiple transitions as an alternative way to create a node level that computes a subset of each level AVC groups and successful results were obtained. Since the RF-Read algorithm does not reorganize the training data, it is necessary to first read a data to find out if this data corresponds to a node calculated in the AVC group. Therefore, in cases where AVC groups do not fit in the main memory, the RF-Read algorithm will show undesirable performance in terms of uptime.

Considering the problems encountered in these two algorithms, a third algorithm is proposed. This new algorithm, which is the result of combining RF-Read and RF-Write algorithms, emerges as a mixed strategy and is called "RF-Hybrid". Basically, the RF-Hybrid algorithm is treated like the RF-Read algorithm in cases where AVC groups belonging to all nodes at a certain current level fit into memory. When this situation cannot be realized, that is, when there is a situation where the AVC groups created in the decision tree of the nodes at the same level do not fit in the existing memory, the RF-Hybrid algorithm switches to the RF-Write algorithm structure as a working system. As a result of the studies on these algorithms, the RF-Hybrid algorithm, which was created as a mixed strategy, was found to be the best performing algorithm compared to other algorithms proposed in RainForest The fourth algorithm created based on the RainForest algorithm is called "RF-Vertical". However, this algorithm is basically designed to be used in situations where even an AVC group cannot be inserted into memory (Gehrke et al., 1998; Gehrke et al., 2000).

3.7. Clouds Algorithm

The CLOUDS algorithm, created by Alsabti, Ranka and Singh in 1998, is a decision tree algorithm based on sampling split points for numerical attributes. This algorithm significantly reduces the computational and input / output complexity of operations performed in large data sets compared to other classifier algorithms, while the decision produced maintains the accuracy values of the trees and the quality of the tree in terms of size. CLOUDS works by evaluating the division points for categorical attributes as in the SPRINT algorithm. However, the evaluation of the separation points for numerical properties is different from other algorithms, it describes the partitioning step for the separation of each internal node.

Basically, two new methods are presented for sampling division points with the Clouds algorithm, that is, for the sampling of the internal nodes in the classification phase. The first method is Sampling of Split points (SS). This method derives the separator from a limited number of split points. The second method is to estimate the Sampling Split Points by Estimation (SSE). This method uses gini values to predict to narrow the search field to find the value that will create the best split and improves SS. It is designed to derive the best or closest separator related to the Gini value. As a common feature of both methods, the gini index can be considered to be evaluated only in a subset of the dividing points along each numeric attribute.

Table 2. Decision Tree Algorithms In Large Data Sets

Algorithm	**Advantages**	**Disadvantages**
SLIQ	- Can work with large datasets and provides good scalability - It performs well in large datasets. - It performs the divisions by evaluating with Gini-Index. - SLIQ is more efficient than SPRINT in cases where class list fits into memory.	- The class list size grows linearly with the number of training data. - SLIQ is well scaled only if there is enough memory for the entire class list available. - SLIQ loses validity for a dataset with more than a million entries.
SPRINT	- It is quite fast compared to Slıq algorithm. - It provides effective parallelization of the division process. - Its continuous qualities enable it to be ranked at once.	- Cannot scale at runtime. - The running time per processor is calculated as O (N). - Cannot scale in memory requirements for the application.
SCALPARC	- Can scale to both runtime and memory requirements. - Continuously only list attributes once. - The overall communication load of the algorithm does not exceed O (N). - Run time per processor does not exceed O (N / p). - The memory required for the application does not exceed O (N / P).	- If the number of processors increases too much, sudden jumps can occur in parallel running time of the algorithm.
RAINFOREST	- It provides 2-5 times faster performance than Sprint algorithm. - It can be applied with all known algorithms. - Uses available main memory to increase efficiency. - Separates scalability problems from decision tree quality problems.	- Requires a minimum main memory to be proportional to the dataset. - In the operating system, it is assumed that the AVC group of the root node fits in the main memory. - The number of training objects and runtime increases linearly for all RainForest algorithms
CLOUDS	- Calculation complexity is lower than other algorithms. - Preserves the accuracy and size of the trees created. - Uses computationally efficient methods to determine dividing points. - Reduces Input / Output requirements.	- Assumes that no errors were made in the sampling process for the splitting operation.

In the Sampling of Split points (SS) method, the range value for each numerical property is divided into specific ranges using a computational technique, and these ranges are assumed to contain approximately the same number of points. The calculated gini index for numerical properties is evaluated within the created range limits. The minimum gini (gini min) values between all range limits of numerical properties are determined. Separation point, where the determined minimum gini value and the value of this strain are calculated, are used as separators. The SS method requires a pass through the dataset to derive the split point.

The Sampling Split Points by Estimation (SSE) method divides each numerical property into specific ranges and finds the minimum gini (gini min) value in the ranges, as in the Sampling of Split points (SS) method. It also estimates the lowest value (gini est) of the gini value for each created range. All ranges whose predicted gini est value is greater than the gini min value are eliminated to obtain a list of potential candidate ranges to be used in division (which can be considered as a kind of pruning). After this process, the gini index is evaluated at each different point in the range in order to determine the separator in a range determined as candidate. This may require another transition over the entire dataset. (Alsabti et al., 1998).

3.8. Sprınt vs Clouds

There are several options to efficiently improve data mining operations in large datasets, the first of which is the use of algorithms that reduce computation and input / output requirements, but essentially perform similar operations with other algorithms. Another option is to use algorithms that reduce computation and input / output requirements by making an approximate estimate of the actual process without causing significant loss of accuracy. CLOUDS is a classification algorithm shown in the second category. As a result of the studies, it has been shown that the CLOUDS algorithm gives as much accurate results as the SPRINT algorithm and has greatly superior calculation features. The approach put forward with the CLOUDS algorithm can provide more attractive alternatives than other classification algorithms, especially when an implicit optimization is required (Shafer et al., 1996; Alsabti et al., 1998).

4. Conclusion

One of the important problems encountered in the data mining process is the difficulties encountered in the classification process in large data sets. Different decision tree structures have been developed to overcome these classification difficulties. In this article, various decision tree algorithms used in large data sets are determined. In accordance with the design goals of decision tree-based classification algorithms, the general structures of these algorithms and their differences were investigated. As a result of this research, it has been seen that there are many practical applications used in classification of big data sets in data mining. It is observed that the studies for processing large data sets and eliminating the problems encountered during the process are still continuing. It can be said that the aim of these new studies is to overcome the shortcomings of the existing algorithms, find fast, effective new ways and create new accuracy tree algorithms with high accuracy.

Referanslar

Akpınar, H. (2000). Veri Tabanlarında Bilgi Keşfi ve Veri Madenciliği, İ. U. işletme Fakültesi Dergisi, C:29, s: 1-22.

Alsabti, K., Ranka, S., Singh, V. (1998). CLOUDS: A Decision Tree Classifier for Large Datasets, Electrical Engineering and Computer Science.Paper 41.

Aytekin, Ç., Sütcü, C.S., Özfidan, U. (2018) Text Classıfıcatıon Vıa Decısıon Trees Algorıthm: Customer Comments Case, The Journal of International Social Research, vol:11 ss:55.

Berry, M.J.A., Lınoff, G.S. (1997). Data Mining Techniques for Marketing, Sales, and Customer Relationship Management, First Edition, Wiley Publishing, 187-216.

Bounsaythip, C., Rinta, R.E. (2001). Overview of Data Mining For Customer Behavior Modeling. VTT Information Technology Research Report. Ver 1. ss. 21.

Breiman, L., Freidman, J.H., Olshen, R.A., Stone, C.J. (1984). Classification and Regression Trees, Monterey, CA: Wadsworth, 358 s.

Chaudhuri, S., Fayyad, U., Berhardt, J. (1997). Scalable Classification over SQL Database, Technical Report MSR-TR-97-35, Microsoft Research.

Chein, C. F., Chen, L. F. (2008) Data Mining to Improve Personnel Selection and Enhance Human Capital: A Case Study in High-Technology Industry, Expert Systems with Applications, vol. 34, p. 280-290.

Çalış, A., Kayapınar, S., Çetinyokuş, T. (2014). Veri Madenciliğinde Karar Ağacı Algoritmaları ile Bilgisayar ve İnternet Güvenliği Üzerine Bir Uygulama, Endüstri Mühendisliği Dergisi cilt:25 sayı:3-4 s:2-19.

Demirel, Ş., Y. Giray, S. (2019). Karar Ağacı Algoritmaları ve Çocuk İşçiliği Üzerine Bir Uygulama. Social Sciences Research Journal, 8 (4), 52-65.

Gehrke, J., Ramakrishnan, R., and Ganti, V. (1998). Rainforest: A framework for fast decision tree construction of large datasets, VLDB, vol. 98, pp. 416–427.

Gehrke, J., Ramakrishnan, R., and Ganti, V. (2000). Rainforest: A framework for fast decision tree construction of large datasets, Data Mining and Knowledge Discovery, 4, 127–162.

Han, J., Kamber, M. (2006). Data Mining: Concepts and Techniques, Morgan Kaufmann, USA.

Hssına, B., Merbouha, A., Ezzıkourı, H., Errıtalı, M. (2014). A comparative study of decision tree ID3 and C4.5, International Journal of Advanced Computer Science and Applications.

Joshi, M. V., Karypis, G., Kumar, V. (1998). ScalParC: A New Scalable and Efficient Parallel Classification Algorithm for Mining Large Datasets, Department of Computer Science University of Minnesota, Minneapolis, MN 55455.

Kavzoğlu, T., Çölkesen, İ. (2010). Classification of Satellite Images Using Decision Trees: Kocaeli Case, Electronic Journal of Map Technologies 2-1, 36-45.

Mehta, M., Agarwal, R. and Rissanen, J. (1996) SLIQ: A fast scalable classifier for data mining, In Proc. of 5th International Conference on Extending Database Technology (EBDT).

Mingers, J. (1989). An empirical comparison of pruning methods for decision tree induction", Machine Learning, 4, 227–243.

Nisbet, R., Elder, J. and Miner, G. (2009), Handbook of Statistical Analysis and Data Mining Applications, Burlington: Elsevier, ISBN: 978-0-12-374765-5.

Oğuzlar, A. (2004). CART Analizi ile Hane Halkı İşgücü Anketi Sonuçlarının Özetlenmesi, Atatürk Üniversitesi İİBF Dergisi, sayı 18, s. 79-90.

Özekeş, S., Çamurcu, A. Y. (2002). Veri Madenciliğinde Sınıflama ve Kestirim Uygulaması, Marmara Üniversitesi Fen Bilimleri Dergisi, sayı 18, s. 1-17.

Patil, N., Lathi, R., Chitre V., (2012), Comparison of C5.0 & CART Classification algorithms using pruning technique, International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 1 Issue 4.

Pehlivan, G. (2006). CHAID Analizi ve Bir Uygulama. Yayınlanmamış Yüksek Lisans Tezi. İstanbul: Yıldız Teknik Üniversitesi, FBE.

Sezer, E. A., Bozkır, A. S., Yağız, S., Gökçeoğlu, C. (2010). Karar Ağacı Derinliğinin CART Algoritmasında Kestirim Kapasitesine Etkisi: Bir Tünel Açma Makinesinin İlerleme Hızı Üzerinde Uygulama, Akıllı Sistemlerde Yenilikler ve Uygulamaları Sempozyumu, Kayseri.

Varçın, F., Erbay, H., and Horasan, F. (2016). Latent semantic analysis via truncated ULV decomposition. 24th Signal Processing and Communication Application Conference (SIU), pp. 1333-1336.

Shafer, J., Agarwal, R. and Mehta, M. (1996) SPRINT: A scalable parallel classifier for data mining, In Proc. of 22nd International Conference on Very Large Databases.

Shannon, C. (1948). A Mathematical Theory of Communication, The Bell System Technical Journal. Vol:27. ss:379-423.

Kacar, S., Eksi, Z., Akgul, A., and Horasan, F. (2013). MATLAB paralel hesaplama araç kutusu ile shannon entropi hesaplanmasi. 1st Internatonal Symposum on Innovatve Technologes in Engneerng and Science, Sakarya, 765-773.

Swain, P.H., Hauska, H. (1977). Decision tree classifier-design and potential, IEEE Transactions on Geoscience and Remote Sensing, 15, 142-147.

Yang, Y., Chen, W. (2016). Taiga: Performance Optimization of the C4.5 Decision Tree Construction Algorithm, Tsınghua Science and Technology, ISSN 1007-0214 06/11, pp 415–425 Vol. 21, N. 4.

Zhixian N., Zong L., Yan, Q., Zhao, Z. (2009), AutoRecognizing DBMS Workload Based on C5.0 Algorithm, College of Computer and Software, Taiyuan University of Technology.