Improved Clustering And Naïve Bayesian Based Binary Decision Tree With Bagging Approach

Medeswara Rao Kondamudi
Mtech Student ,VVIT

Sudhir Tirumalasetty
CSE , Assoc Prof , VVIT

Abstract –

Decision Trees provide an attractive classification scheme which is responsible for making reliable decisions and possibly interpret them. Bayesian averaging over Decision trees allows estimating on attributes to assess the class posterior distribution and estimates the chance of making misleading decisions. The clustering problem has actually been addressed in several contexts in plenty of disciplines; due to this problem experimental data needs to clean the data before applying the data mining techniques. In this paper a new framework is proposed by integrating decision tree based attribute selection for data clustering.

***Keywords* – Decision tree, Classifier, NaiveBayes.**

I. INTRODUCTION

The quantification of uncertainty in decisions is of crucial important for medical systems. Bayesian averaging over decision models allows clinicians to evaluate the class posterior distribution and therefore to estimate the risk of making misleading decisions. The use of Decision Trees (DTs) allows experts to understand how decisions are made. Data analysis procedures can easily be dichotomized as either exploratory or confirmatory, driven by availability of appropriate models for your data source, but a key element in each of varieties of procedures (whether for hypothesis formation or decision-making) is the grouping, or classification of measurements in accordance to either (i) goodness-of-fit to the postulated model, or (ii) natural groupings (clustering) revealed through analysis. Cluster analysis is the organization of a variety of patterns (usually represented to be the vector of measurements, or possibly a point inside a multidimensional space) into clusters based upon similarity. Intuitively, patterns in the context of a valid cluster are usually more similar to each other than they're to a pattern part of a distinct cluster.
Clustering is a vital method in data warehousing and data mining. It groups similar object together within a cluster (or clusters) and dissimilar object in other cluster (or clusters) or remove from the clustering process. That really is, in fact its's an unsupervised classification in data analysis that arises in many applications in numerous fields such as data mining[3], image processing, machine learning and bioinformatics. Since, in fact its's an unsupervised learning method, it does not need train datasets and pre-defined taxonomies. Fact is that there are several special requirements for search result pages clustering algorithms, two of which most important is, clustering performance and meaningful cluster description. Plenty of clustering technique is available, among those hierarchical clustering and Partition Clustering happens to be the widely used clustering methods. A Partition-clustering algorithm with their outputs produce one clustering set that involves disjoint clusters, i.e., the comprehensive data description is flat. Basically, partitioned clustering is nothing but pre-defined large number of partition range. In which the total number of partition (k) range should be less than number of object (n) among the dataset. Partition clustering always should satisfy the condition $k < n$.
A Hierarchical clustering is naturally a nested of partitions technique depend on the business requirements. It produces not merely one clustering taking place in their outputs except a hierarchy of clusters. This procedure function for both type of approach either bottom up and top down approach. Within this method all record object arranged between a huge cluster, then big cluster are continuously divided into small clusters.

There are actually mainly two machine learning strategies [2]:
1) Supervised learning
In supervised learning, the system is supplied with the appropirate discuss each training example. The work of one's system is to discover this relationship connecting the input examples, and of course the answers.
For instance, a system just might be shown various images of faces, where each one has domain. The machine could then be shown a different image perhaps one of the faces, and will output the name of a given face.
2) Unsupervised Learning Strategy
In unsupervised learning, the operating system is not provided with any answers, or correct outputs. The academic process usually aims to locate patterns and correlations within the data. For instance, a store could

record the items that people buy; a learning system could then find correlations between different items that may bought together.

Decision tree induction [1], will be the learning of decision trees from class-labeled training tuples. A call tree serves as a flowchart-like tree structure, where each internal node (nonleaf node) denotes a test traveling on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label. The topmost node inside a tree is the root node. Instances are classified beginning from the main node and sorted dependent on their feature values. The leaf node reached is regarded as the instruction label for that example. The algorithm can naturally handle binary or multiclass classification problems. The leaf nodes can refer to either of the K classes concerned.

Fig. 1 shows an example of a decision tree for the training set of Table 1.
The feature that best divides the training data would be the root node of the tree. The same procedure is then repeated on each partition of the divided data, creating subtrees until the training data is divided into subsets of the same class[1].

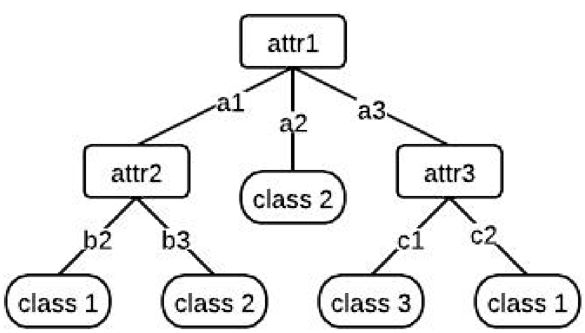


Fig. 1: An example decision tree

AN EXAMPLE DATA SET
attr1 attr2 attr3 Class
a1 b2 c3 1
a2 b1 c2 2
a3 b1 c1 3
a2 b1 c3 2
a1 b2 c2 1
a3 b3 c1 3
a1 b3 c2 2
a1 b2 c1 1
a3 b3 c2 1

1. Create a root node for the tree that best classifies examples
2. A ← best decision attribute for next node
a. assign A as decision attribute for the node
b. for each value of A, create new descendant of node
c. sort training examples to leaf node
d. if training examples are perfectly classified then stop else, iterate over new leaf nodes.

Apart from hierarchical
binary classifiers, two popular techniques for binary classification have also been used: One-Versus-All (OVA)
and One-Versus-One (OVO). In addition, we utilize a method of using all binary classifiers to observe the performance of the hierarchical binary classifier. Since building all hierarchical binary classifiers is computation intensive, we propose a greedy technique for building hierarchical binary classifiers. Feature selection is an important step in the design of a classification system. Selecting a good subset of features decreases computational load and can also improve accuracy. Including random or noisy features can cause classifiers to learn incorrect associations.

II. LITERATURE SURVEY

Multiclass classification problem would be to map information samples into a little more than two classes. There's only two main approaches for solving multiclass classification problems. The first approach deals directly with the multiclass problem and uses algorithms like Decision Trees, Neural Networks [1], k-Nearest Neighbor and Naive Bayesian classifiers. The main trouble with this method is to determine features which can distinguish classes when the wide range of classes increases. Consequently, this procedure is likely to yield lower accuracy. A classification algorithm for data streams must meet several different requirements from the traditional setting (Bifet et al., 2009). The most significant are the following. First, process one example at a time, and inspect it at most once. The data examples flow in and out of a system one after another. Each example must be accepted in the order in which it arrives. Once inspected or ignored, the example is discarded with no way to retrieve it. Second, use a limited amount of memory. Memory will be easily exhausted without limiting its allocation since the amount of the data is potentially infinite. Third, work in a limited amount of time. Though most conventional algorithms are fast enough when classifying examples, the training processes are time consuming. For an algorithm to scale comfortably to any number of examples, its training complexity must be linear to the number of examples, such that online learning is possible. Fourth, be ready to perform classification at any time. This is the so-called any-time property, which indicates that the induction model is ready to be applied at any point between training examples[1].

Decision tree induction algorithms on data streams
Decision tree is one of the most often used techniques in the data mining literature. Each node of a decision tree contains a test on an attribute. Each branch from a node corresponds to a possible outcome of the test and each leaf contains a class prediction. A decision tree is constructed by recursively
replacing leaves by test nodes, starting at the root. The

attribute to test in a leaf is chosen by comparing all available attributes and choosing the best one[2]. according to some heuristic evaluation function. Classic decision tree learners like ID3, C4.5, and CART assume that all training examples can be stored simultaneously in memory, and thus are severely limited in the number of examples from which they can learn.

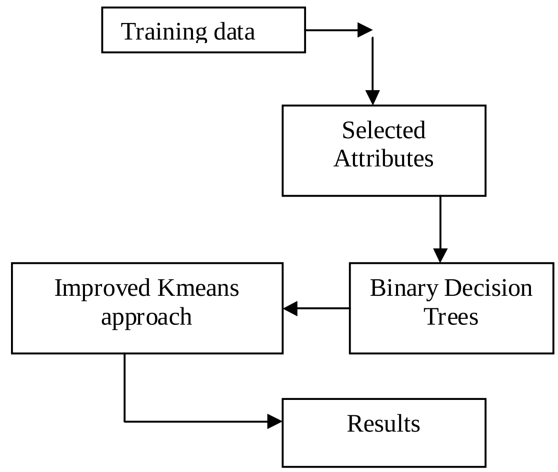
Predictive clustering

In particular, the predictive modeling methods that partition the examples into subsets, e.g., decision trees and decision rules, can also be viewed as clustering methods.Namely, a decision tree can be regarded as a hierarchy of clusters, where each node is a cluster; such a tree is called a clustering tree. Likewise, a decision rule can represent a cluster of examples which it covers. The benefit of using these methods for clustering is that, in addition to the clusters themselves, we also get symbolic descriptions of the constructed clusters. Every cluster in a tree has a symbolic description in the form of a conjunction of conditions on the path from the root of the tree to the given node, and every cluster represented by a rule is described by the rule's condition. There is, however, a difference between 'tree' clusters and 'rule'[4] clusters. 'Tree' clusters are ordered in a hierarchy and do not overlap, while 'rule' clusters in general are not ordered in any way (they are flat) and can overlap (one example can belong to more than one cluster). We can say that clustering trees are a hierarchical clustering method, and clustering rules are a partitional (and possibly fuzzy) clustering method.

Brieman, Friedman, Olshen, and Stone developed the CART algorithm in 1984. It builds a binary tree. Observations are split at each node by a function on one attribute. The split is selected which divides the observations at a node into subgroups in which a single class most predominates. When no split can be found that increases the class specificity at a node the tree has reached a leaf node. When all observations are in leaf nodes the tree has stopped growing. Each leaf can then be assigned a class and an error rate (not every observation in a leaf node is of the same class). Because the later splits have smaller and less representative samples to work with they may overfit the data. Therefore, the tree may be cut back to a size which allows effective generalization to new data. Branches of the tree that do not enhance predictive classification accuracy are eliminated in a process known as "pruning."

III. PROPOSED SYSTEM

The major advantage of the Naïve Bayes classifier is that it requires short time for training the classifier. Also, each training example has an effect on the prediction and each training example in turn would increase/decrease the probability that a prediction is correct. However, the assumption of independence among attributes is not true always and hence the accuracy of Naïve Bayes classifier is unstable.



DATASET FILE FORMATS

In this project two types of file formats are used. They are

i) CSV

ii) ARFF

i. **CSV**: It stands for Comma Separated Value. This format is obtained using MS-Excel. KDD99 dataset is loaded into Excel and then it is saved with an extension of csv.

ii. **ARFF**: It stands for Attribute Relation File Format. An file is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files were developed by the Machine Learning Project at the Department of Computer Science of The University of Waikato for use with the Weka machine learning software

ARFF files have two distinct sections. The first section is the Header information, which is followed the Data in The ARFF Header Section

The ARFF Header section of the file contains the relation declaration and attribute declarations.

The @relation Declaration

The relation name is defined as the first line in the ARFF file. The format is:

@relation <relation-name>

where <relation-name> is a string. The string must be quoted if the name includes spaces.

The @attribute Declarations:

Attribute declarations take the form of an orderd sequence of @attribute statements. Each attribute in the data set has its own @attribute statement which uniquely defines the name of that attribute and it's data type. The order the attributes are declared indicates the column position in the data section of the file. For example, if an attribute is the third one declared then Weka expects that all that attributes values will be found in the third comma delimited column.

The format for the @attribute statement is:

@attribute <attribute-name> <datatype>

where the <attribute-name> must start with an alphabetic character. If spaces are to be included in the name then the entire name must be quoted.

The <datatype> can be any of the four types currently (version 3.2.1) supported by Weka:

- numeric
- <nominal-specification>
- string
- date [<date-format>]

where <nominal-specification> and <date-format> are defined below. The keywords numeric, string and date are case insensitive.

Nominal attributes

Nominal values are defined by providing an <nominal-specification> listing the possible values: {<nominal-name1>, <nominal-name2>, <nominal-name3>, ...}

String attributes

String attributes allow us to create attributes containing arbitrary textual values. String attributes are declared as follows:

@ATTRIBUTE name string

Decision tree is a tree structure, where internal nodes denote a test on an attribute, each branch represents the outcomes of the test and the leaf node represents the class labels. Decision tree induction is the learning of decision trees from class-labeled training tuples. Construction of decision trees is simple and fast, and does not need any domain knowledge and hence appropriate for exploratory knowledge discovery. In general, decision tree classifiers have good accuracy, but successful use of it depends on the data at hand. Decision trees are used for classification and classification rules are easily generated from them. An unknown tuple X can be classified, given its attribute values by testing the attribute values against the decision tree. The general decision tree algorithm takes the training data set, attribute list and attribute selection method as input. The algorithm creates a node, and then applies attribute selection method to determine the best splitting criteria and the created node is named by that attribute. Subset of training tuples is formed using the splitting attribute. The algorithm is called recursively for each subset, till the subset contains tuples of same class. When the subset contains tuples from the same class a leaf is attached with a label of the majority class in the training set from the root. ID3, C4.5, and CART adopt a greedy, non-backtracking approach in which decision trees are constructed in a top-down recursive divide-and-conquer.

Feature Selection

A preprocessing technique feature selection identifies and removes irrelevant attributes that do not play any role in the classification task. Several feature selection methods are available with different search techniques to produce a reduced data set. This reduced data set improves accuracy compared with original dataset. Feature selection does not alter the relevance or meaning the data set. The feature selection methods are categorized as filter, wrapper and hybrid. The result of these methods varies in time and accuracy.

ROBUST BAGGING ALGORITHM:

A bagging algorithm for multiple classification into several classes.
1 Initialisation of the training set D
2 for m = 1, ..., M
2.1 Creation of a new set Dm of the same size D by random selection of training examples from the set D (some of examples can be selected repeatedly and some may mot be selected at all).
2.2 Learning of a particular classifier Hm: Dm → R by a given machine learning algorithm based on the actual training set Dm.
3. Compound classifier H is created as the aggregation of particular classifiers Hm: m = 1, ...,M and an example di is classified to the class cj in accordance with the number of votes obtained from particular classifiers Hm.

$$H(d_i, c_j) = sign\left(\sum_{m=1}^{M} \alpha_m H_m(d_i, c_j)\right)$$

If it is possible to influence the learning procedure performed by the classifier *Hm* directly, classification error can be minimised also by *Hm* while keeping parameters *αm* constant.
step 4: For each continuous attributes in D find the each

adjacent pair of continuous attribute
values that are not classified into the same class value for that continuous attribute
Step 5: Calculate the prior probabilities P(Cj) and conditional probabilities P(Aij|Cj) in D.
Step 6: Classify all the training examples using these prior and conditional probabilities, P(ei
|
Cj) = P(Cj) ∏k=1→p P(Aij | Cj).
Step 7: Update the class value for each example in D with Maximum Likelihood (ML) of posterior probability,
P(Cj|ei); Cj = Ci→ PML(Cj|ei).
Step 8: Recalculate the prior P(Cj) and conditional P(Aij|Cj) probabilities using updated class values in D.

Cluster analysis

Clustering finds groups of similar data either by partitioning data into *k* subsets (partitioning methods) or creating a hierarchical decomposition of the data (hierarchical methods) or building groups of data based on density, Hierarchical clustering creates a hierarchy of data records and subsets either by dividing the whole data set until a given size of subsets (divisive approach) or agglomerating records into subsets until all records belong to a given number of sets or one set (agglomerative approach). Division and merging is based on the distance between groups called linkage that can be calculated as the distance between closest or center points of groups as well as distance among all points of different groups

IMPROVED KMEANS:

The pseudo code for the **adapted kMean algorithm** is presented as below:

1. Choose random k data points as initial Clusters Mean (cluster center)

2. Repeat

3. for each data point x from D

4. Computer the distance x and each cluster mean (centroid)

5. Assign x to the nearest cluster.

6. End for

7. Re-compute the mean for current cluster collections.

8. Until reaching stable cluster

9. Use these centroid for normal and outlier.

10. Calculate distance of centroid from normal and outlier centroid points.

11. If distance(X, Dj) > = thres

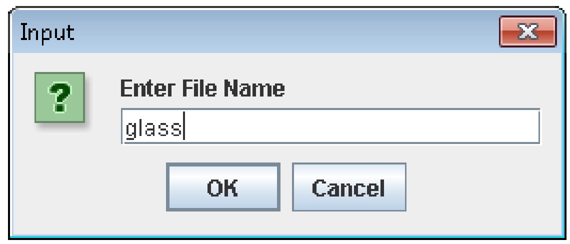
12. Then outlier found ; exit

13. Else then

14. X is normal;

IV.EXPERIMENTAL RESULTS

All experiments were performed with the configurations Intel(R) Core(TM)2 CPU 2.13GHz, 2 GB RAM, and the operating system platform is Microsoft Windows XP Professional (SP2).



Selected attributes: 1,2,3,4,6,7,8,9 : 8
RI
Na
Mg
Al
K
Ca
Ba
Fe

All the best NBDTrees:

IMPROVED BINARY DECISION TREE USING NAIVE BAYES AND CLUSTERING TREE

```
Ba <= 0.27
|  Mg <= 2.41
|  |  K <= 0.11
|  |  |  RI <= 1.52068: tableware (11.0)
|  |  |  RI > 1.52068: build wind non-float (4.0)
|  |  K > 0.11
|  |  |  Na <= 13.49: containers (14.0/1.0)
|  |  |  Na > 13.49: build wind non-float (5.0/1.0)
|  Mg > 2.41
|  |  Al <= 1.41
|  |  |  RI <= 1.51689
|  |  |  |  Fe <= 0.17: vehic wind float (7.0/1.0)
|  |  |  |  Fe > 0.17: build wind float (2.0)
|  |  |  RI > 1.51689
|  |  |  |  K <= 0.23
|  |  |  |  |  Mg <= 3.56: build wind non-float (2.0/1.0)
```

```
| | | | |  Mg > 3.56
| | | | | |  RI <= 1.52127
| | | | | | |  RI <= 1.52101: build wind float (5.0/1.0)
| | | | | | |  RI > 1.52101: vehic wind float (3.0)
| | | | | |  RI > 1.52127: build wind float (16.0/1.0)
| | | |  K > 0.23
| | | | |  RI <= 1.518: build wind float (34.0/2.0)
| | | | |  RI > 1.518
| | | | | |  Mg <= 3.6
| | | | | | |  RI <= 1.52119: build wind float
(11.0/1.0)
| | | | | | |  RI > 1.52119: headlamps (2.0)
| | | | | |  Mg > 3.6: build wind non-float (16.0)
| |  Al > 1.41
| | |  Ca <= 8.89
| | | |  Fe <= 0.16: build wind non-float (39.0/1.0)
| | | |  Fe > 0.16
| | | | |  Ba <= 0.0
| | | | | |  Na <= 12.87: build wind float (4.0)
| | | | | |  Na > 12.87: build wind non-float (3.0/1.0)
| | | | |  Ba > 0.0: build wind non-float (4.0)
| | |  Ca > 8.89: vehic wind float (4.0/1.0)
Ba > 0.27: headlamps (28.0/2.0)
```

Number of Leaves : 20

Size of the tree : 39

Correctly Classified Instances 207 96.729 %
Incorrectly Classified Instances 7 3.271 %
Mean absolute error 0.0471
Relative absolute error 22.2462 %
Total Number of Instances 214

IRIS DATASET:

Selected attributes: 3,4 : 2
petallength
petalwidth

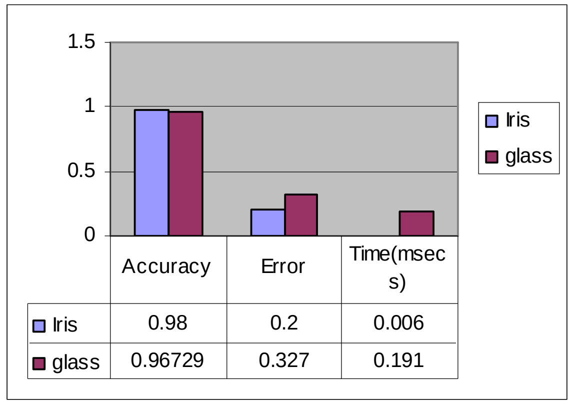
All the best NBDTrees:

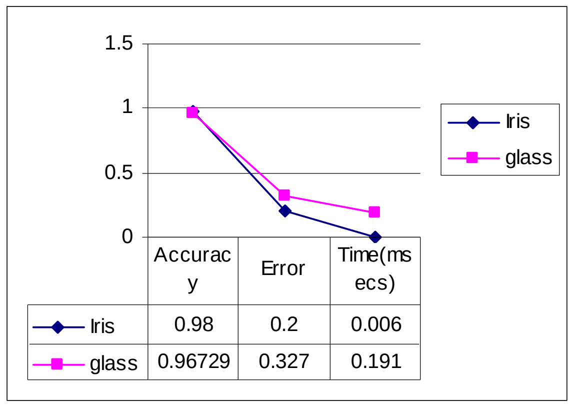
IMPROVED BINARY DECISION TREE USING NAIVE BAYES AND CLUSTERING TREE

```
petalwidth <= 0.6: Iris-setosa (54.0)
petalwidth > 0.6
|  petalwidth <= 1.7
| |  petallength <= 4.9: Iris-versicolor (48.0/1.0)
| |  petallength > 4.9
| | |  petalwidth <= 1.5: Iris-virginica (4.0)
| | |  petalwidth > 1.5: Iris-versicolor (3.0/1.0)
|  petalwidth > 1.7: Iris-virginica (41.0)
```

Number of Leaves : 5

Size of the tree : 9
Correctly Classified Instances 147 98 %
Incorrectly Classified Instances 3 2 %
Mean absolute error 0.0272
Relative absolute error 6.1228 %
Total Number of Instances 150





Above two graphs shows that accuracy,error and time of execution to different datasets.

V. CONCLUSION AND FUTURE SCOPE

In this paper, we analyzed on the construction of binary decision tree model using naïve bayes and bagging approach, and clustering method. We propose a new condition of generating terminal nodes so that the decision tree is optimized in the number of nodes and leaves. In addition, the speed is also improved. But when dataset has small number of attributes, the precision will be influenced.Experimental results give better results appro > 96 % with different datasets are tested. The pruning method of this algorithm will be also studied in future work.

REFERENCES

[1] An Empirical Analysis of Multiclass Classification Techniques in Data Mining Radhika Kotecha, Vijay Ukani and Sanjay Garg.

[2] W. R. Smart, Genetic programming for multiclass object classification," Master's thesis, Victoria University of Wellington, New Zealand, 2005.

[3] Quinlan JR, "Simplifying decision trees," J. International Journalof ManMachine Studies, vol.27, pp. 221-234, 1987

[4] Quinlan JR, "Learning efficient classification procedures and their application to chess and games[A]," Machine Learning: an artificial intelligence approach [C]. San Mateo, CA: Morgan Kaufmann, 1983, pp. 463-48

[5] Minos Garofalakix. DongJoon Hyun. Rajeev Rastogi. Kyuseok Shim, "Building Decision Trees with Constraints," Data Mining and Knowledge Discovery, vol.27,pp. 187–214 2003

[6] Wang Xizhao,Yang Chenxiao,"Merging-Branches Impact on Decision Tree Induction," Journal of Computers, vol.30 No.8,pp. 1251-1258, 2007(Chinese)

[7] Zhou Yatong,Zhang Taiyi,Luzhaogan, "Decision Tree Model Based on Bayesian Inferenc," Journal of Xi an Jiaotong University, vol.40, pp. 888-891, 2006(chinese)4