ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 5, Issue 5, May 2016

Classification Algorithms of Data Mining

Bharat S. Makhija, Dr. Anjali B. Raut

Abstract- **Data mining is sorting through data to identify patterns and establish relationships. Data mining parameters include: Association - looking for patterns where one event is connected to another event, Sequence or path analysis - looking for patterns where one event leads to another later event, Classification - looking for new patterns (May result in a change in the way the data is organized but that's ok), Clustering - finding and visually documenting groups of facts not previously known, Forecasting - discovering patterns in data that can lead to reasonable predictions about the future (This area of data mining is known as predictive analytics). Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks.**

Index Terms**- Decision tree induction classification, Naïve-Bayesian classification, Rule-Based classification, k-nearest neighbor classification**

I. INTRODUCTION

A classification task begins with a data set in which the class assignments are known. For example, a classification model that predicts credit risk could be developed based on observed data for many loan applicants over a period of time. In addition to the historical credit rating, the data might track employment history, home ownership or rental, years of residence, number and type of investments, and so on. Credit rating would be the target, the other attributes would be the predictors, and the data for each customer would constitute a case.

Classifications are discrete and do not imply order. Continuous, floating-point values would indicate a numerical, rather than a categorical, target. A predictive model with a numerical target uses a regression algorithm, not a classification algorithm. The simplest type of classification problem is binary classification. In binary classification, the target attribute has only two possible values: for example, high credit rating or low credit rating. Multiclass targets have more than two values: for example, low, medium, high, or unknown credit rating.

In the model build (training) process, a classification algorithm finds relationships between the values of the predictors and the values of the target. Different classification algorithms use different techniques for finding relationships. These relationships are summarized in a model, which can then be applied to a different data set in which the class assignments are unknown.

II. CLASSIFICATION ALGORITHMS

A. Decision tree induction classification:

During the late 1970s and early 1980s, J. Ross Quinlan, a researcher in machine learning, developed a decision tree algorithm known as ID3 (Iterative Dichotomiser). The Algorithm is as follows :

Algorithm:

Generate decision tree. Generate a decision tree from the training tuples of datapartition D.

Input:

1. Data partition, D, which is a set of training tuples and their associated class labels;

2. attribute list, the set of candidate attributes;

3. Attribute selection method, a procedure to determine the splitting criterion that “best” partitions the data tuples into individual classes. This criterion consists of a splitting attribute and, possibly, either a split point or splitting subset.

Output:

A decision tree.

Method:

(1) create a node N;
(2) if tuples in D are all of the same class, C then
(3) return N as a leaf node labeled with the class C;
(4) if attribute list is empty then
(5) return N as a leaf node labeled with the majority class in D; // majority voting
(6) apply Attribute selection method(D, attribute list) to find the “best” splitting criterion;
(7) label node N with splitting criterion;
(8) if splitting attribute is discrete-valued and multiway splits allowed then // not restricted to binary trees
(9) attribute list attribute list splitting attribute; // remove splitting attribute
(10) for each outcome j of splitting criterion
// partition the tuples and grow subtrees for each partition
(11) let Dj be the set of data tuples in D satisfying outcome j; // a partition
(12) if Dj is empty then

(13) attach a leaf labeled with the majority class in D to node N;

(14) else attach the node returned by Generate decision tree(Dj, attribute list) to node N;

endfor

(15) return N;

The strategy of algorithm is as follows:

1. The algorithm is called with three parameters: D, attribute list, and Attribute selection method.We refer to D as a data partition. Initially, it is the complete set of training tuples and their associated class labels. The parameter attribute list is a list of attributes describing the tuples. Attribute selection method specifies a heuristic procedure for selecting the attribute that "best" discriminates the given tuples according to class. This procedure employs an attribute selection measure, such as informationgain or the gini index. Whether the tree is strictly binary is generally driven by the attribute selection measure. Some attribute selection measures, such as the gini index, enforce the resulting tree to be binary. Others, like information gain, do not, therein allowing multiway splits (i.e., two or more branches to be grown from a node).

2. The tree starts as a single node, N, representing the training tuples in D (step 1).

3. If the tuples in D are all of the same class, then node N becomes a leaf and is labeled with that class (steps 2 and 3). Note that steps 4 and 5 are terminating conditions. All of the terminating conditions are explained at the end of the algorithm. 4. Otherwise, the algorithm calls Attribute selection method to determine the splitting criterion. The splitting criterion tells us which attribute to test at node N by determining the "best" way to separate or partition the tuples in D into individual classes (step 6). The splitting criterion also tells us which branches to grow from node N with respect to the outcomes of the chosen test. More specifically, the splitting criterion indicates the splitting attribute and may also indicate either a split-point or a splitting subset. The splitting criterion is determined so that, ideally, the resulting partitions at each branch are as "pure" as possible. A partition is pure if all of the tuples in it belong to the same class. In other words, if we were to split up the tuples in D according to the mutually exclusive outcomes of the splitting criterion, we hope for the resulting partitions to be as pure as possible.

5. The node N is labeled with the splitting criterion, which serves as a test at the node (step 7). A branch is grown from node N for each of the outcomes of the splitting criterion. The tuples in D are partitioned accordingly (steps 10 to 11). There are three possible scenarios, as illustrated in Figure 6.4. Let A be the splitting attribute. A has v distinct values, {a1, a2, : : : , av}, based on the training data.

A is discrete-valued: In this case, the outcomes of the test at node N correspond directly to the known values of A. A branch is created for each known value, aj, of A and labeled with that value (Figure 6.4(a)). Partition Dj is the subset of class-labeled tuples in D having value aj of A. Because all of the tuples in a given partition have the same value for A, then A need not be considered in any future partitioning of the tuples. Therefore, it is removed from attribute list(steps 8 to 9).

A is continuous-valued: In this case, the test at node N has two possible outcomes, corresponding to the conditions A _ split point and A > split point, respectively, where split point is the split-point returned by Attribute selection method as part of the splitting criterion. (In practice, the split-point, a, is often taken as the midpoint of two known adjacent values of A and therefore may not actually be a pre-existing value of A from the training data.) Two branches are grown from N and labeled according to the above outcomes (Figure 6.4(b)). The tuples are partitioned such thatD1 holds the subset of class-labeled tuples in D for which A_split point,while D2 holds the rest.

A is discrete-valued and a binary treemust be produced (as dictated by the attribute selection measure or algorithm being used): The test at node N is of the form "A € SA?". SA is the splitting subset for A, returned by Attribute selection method as part of the splitting criterion. It is a subset of the known values of A. If a given tuple has value aj of A and if aj € SA, then the test at node N is satisfied. Two branches are grown from N (Figure 6.4(c)). By convention, the left branch out of N is labeled yes so that D1 corresponds to the subset of class-labeled tuples in Dthat satisfy the test. The right branch out of N is labeled no so that D2 corresponds to the subset of class-labeled tuples from D that do not satisfy the test.

6. The algorithm uses the same process recursively to form a decision tree for the tuples at each resulting partition, Dj, of D (step 14).

7. The recursive partitioning stops only when any one of the following terminating conditions is true:

All of the tuples in partition D (represented at node N) belong to the same class (steps 2 and 3), or

There are no remaining attributes on which the tuples may be further partitioned (step 4). In this case, majority voting is employed (step 5). This involves converting node N into a leaf and labeling it with the most common class in D. Alternatively, the class distribution of the node tuples may be stored.

There are no tuples for a given branch, that is, a partition Dj is empty (step 12).

In this case, a leaf is created with the majority class in D (step 13).

8. The resulting decision tree is returned (step 15).

B. Naïve Bayesian Classification:

The naïve Bayesian classifier, or simple Bayesian classifier, works as follows:

1. Let D be a training set of tuples and their associated class labels. As usual, each tuple is represented by an n-dimensional attribute vector, **X** = (x1, x2, : : : , xn), depicting n measurements made on the tuple from n attributes, respectively, A1, A2, : : : , An.
2. Suppose that there are m classes, C1, C2, : : : , Cm. Given a tuple, **X**, the classifier will predict that **X** belongs to the class having the highest posterior probability, conditioned on **X**. That is, the naïve Bayesian classifier predicts that tuple **X** belongs to the class Ci if and only if

$$P(C_i|X) > P(C_j|X) \quad \text{for } 1 \le j \le m, j \ne i.$$

Thus we maximize P(Ci /**X**). The classCi for which P(Ci / **X**) is maximized is called the maximum posteriori hypothesis. By Bayes' theorem

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}.$$

3. As P(X) is constant for all classes, only P(X / Ci) P(Ci) need be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, P(C1) = P(C2) = - - - = P(Cm), and we would therefore maximize P(X / Ci). Otherwise, we maximize P(X / Ci) P(Ci).
4. Given data sets with many attributes, it would be extremely computationally expensive to compute P(X / Ci). In order to reduce computation in evaluating P(X / Ci), the naive assumption of class conditional independence is made. This presumes that the values of the attributes are conditionally independent of one another, given the class label of the tuple (i.e., that there are no dependence relationships among the attributes). Thus,

$$\begin{aligned} P(X|C_i) &= \prod_{k=1}^{n} P(x_k|C_i) \\ &= P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_n|C_i). \end{aligned}$$

We can easily estimate the probabilities P(x1 / Ci), P(x2 / Ci), : : : , P(xn / Ci) from the training tuples. Recall that here xk refers to the value of attribute Ak for tuple **X**. For each attribute, we look at whether the attribute is categorical or continuous-valued.
5. In order to predict the class label of **X**, P(**X** / Ci) P(Ci) is evaluated for each class Ci. The classifier predicts that the class label of tuple **X** is the class Ci if and only if

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j) \quad \text{for } 1 \le j \le m, j \ne i.$$

In other words, the predicted class label is the class Ci for which P(**X** / Ci) P(Ci) is the maximum. Various empirical studies of this classifier in comparison to decision tree and neural network classifiers have found it to be comparable in some domains. In theory, Bayesian classifiers have the minimum error rate in comparison to all other classifiers. However, in practice this is not always the case, owing to inaccuracies in the assumptions made for its use, such as class conditional independence, and the lack of available probability data.
Bayesian classifiers are also useful in that they provide a theoretical justification for other classifiers that do not explicitly use Bayes' theorem. For example, under certain assumptions, it can be shown that many neural network and curve-fitting algorithms output the maximum posteriori hypothesis, as does the naïve Bayesian classifier.

C. Rule-Based Classification:
In this section, we look at rule-based classifiers, where the learned model is represented as a set of IF-THEN rules. We first examine how such rules are used for classification. We then study ways in which they can be generated, either froma decision tree or directly from the training data using a sequential covering algorithm.
IF-THEN rules can be extracted directly from the training data (i.e., without having to generate a decision tree first) using a sequential covering algorithm. The name comes from the notion that the rules are learned sequentially (one at a time), where each rule for a given class will ideally cover many of the tuples of that class (and hopefully none of the tuples of other classes). Sequential covering algorithms are the most widely used approach to mining disjunctive sets of classification rules, and form the topic of this subsection. Note that in a newer alternative approach, classification rules can be generatedusing associative classification algorithms, which search for attribute-value pairs that occur frequently in the data. These pairs may form association rules, which can be analyzed and used in classification.
Basic sequential covering algorithm is as follows:

Algorithm:
Sequential covering. Learn a set of IF-THEN rules for classification.
Input:
I. D, a data set class-labeled tuples;
II. Att vals, the set of all attributes and their possible values.
Output:
A set of IF-THEN rules.
Method:
(1) Rule set = { }; // initial set of rules learned is empty
(2) for each class c do
(3) repeat
(4) Rule = Learn One Rule(D, Att vals, c);
(5) remove tuples covered by Rule from D;
(6) until terminating condition;
(7) Rule set = Rule set +Rule; // add new rule to rule set
(8) endfor
(9) return Rule Set;

D. k-Nearest-Neighbor Classifiers:
The k-nearest-neighbor method was first described in the early 1950s. The method is labor intensive when given large training sets, and did not gain popularity until the 1960s when increased computing power became available. It has since been widely used in the area of pattern recognition.
Nearest-neighbor classifiers are based on learning by analogy, that is, by comparing a given test tuplewith training tuples that are similar to it. The training tuples are described by n attributes. Each tuple represents a point in an n-dimensional space. In this way,
all of the training tuples are stored in an n-dimensional pattern space. When given an unknown tuple, a k-nearest-neighbor

classifier searches the pattern space for the k training tuples that are closest to the unknown tuple. These k training tuples are the k "nearest
neighbors" of the unknown tuple.

"Closeness" is defined in terms of a distance metric, such as Euclidean distance. The Euclidean distance between two points or tuples, say, X1 = (x11, x12, : : : , x1n) and X2 = (x21, x22, : : : , x2n), is

$$dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}.$$

In other words, for each numeric attribute, we take the difference between the corresponding values of that attribute in tuple X1 and in tuple X2, square this difference, and accumulate it. The square root is taken of the total accumulated distance count. Typically, we normalize the values of each attribute before using Equation.

For k-nearest-neighbor classification, the unknown tuple is assigned the most common class among its k nearest neighbors. When k = 1, the unknown tuple is assigned the class of the training tuple that is closest to it in pattern space. Nearestneighbor classifiers can also be used for prediction, that is, to return a real-valued prediction for a given unknown tuple. In this case, the classifier returns the average value of the real-valued labels associated with the k nearest neighbors of the unknown tuple.

III. CONCLUSION

The goal of classification algorithms is to generate more certain, precise and accurate system results. Numerous methods have been suggested for the creation of ensemble of classifiers. Classification methods are typically strong in modeling interactions. Several of the classification methods produce a set of interacting logic that best predict the phenotype. However, a straightforward application of classification methods to large numbers of markers has a potential risk picking up randomly associated markers. But still it is difficult to recommend any one technique as superior to others as the choice of a dataset. Finally, there is no single classification algorithms is best for all kind of dataset. Classification algorithms are specific in their problem domain.

REFERENCES

[1] G.Kesavaraj, Dr.S.Sukumaran "A Study On Classification Techniques in Data Mining" 4th ICCCNT - 2013 Tiruchengode, India July 4 - 6, 2013.

[2] Sundar.C, M.Chitradevi and Dr.G.Geetharamani "Classification of Cardiotocogram Data using Neural Network based Machine Learning Technique" International Journal of Computer Applications (0975 – 888) Volume 47– No.14, June 2012.

[3] Smitha .T, V. Sundaram "Comparative Study Of Data Mining Algorithms For High Dimensional Data Analysis" International Journal Of Advances In Engineering & Technology, Sept 2012.IJAET ISSN: 2231-1963

[4] K Priya, R. Geetha Ramani and Shomona Gracia Jacob "Data Mining Techniques for Automatic recognition of Carnatic Raga Swaram notes" International Journal of Computer Applications (0975 – 8887) Volume 52– No.10, August 2012

[5] Bendi Venkata Ramana1, Prof. M.Surendra Prasad Babu2, Prof. N. B. Venkateswarlu "A Critical Study of Selected Classification Algorithms for Liver Disease Diagnosis" International Journal of Database Management Systems (IJDMS), Vol.3, No.2, May 2011

[6] Koliastasis C and D.K. Despotis "Rules for Comparing Predictive Data Mining Algorithms by Error Rate" OPSEARCH, VOL. 41, No. 3, 2004 Operational Research Society of India

[7] Matthew N. Anyanwu, Sajjan G. Shiva "Comparative Analysis of Serial Decision Tree Classification Algorithms" International Journal of Computer Science and Security, (IJCSS) Volume (3) : Issue (3) page number 230

[8] Mrs.P.Nancy, R. Geetha Ramani and Shomona Gracia Jacob "A Comparison on Performance of Data Mining Algorithms in Classification of Social Network Data" International Journal of Computer Applications (0975 – 8887) Volume 32– No.8, October 2011

[9] P. Bhargavi 1, Dr. S. Jyothi "Soil Classification Using Data Mining Techniques: A Comparative Study" International Journal of Engineering Trends and Technology- July to Aug Issue 2011

[10] Aman Kumar Sharma Suruchi Sahni "Comparative Study of Classification Algorithms for Spam Email Data Analysis" International Journal on Computer Science and Engineering (IJCSE) ISSN : 0975-3397 Vol. 3 No. 5 May 2011 1890-1895.

[11] Thair Nu Phyu "Survey of Classification Techniques in Data Mining" Proceedings of the International Multi Conference of Engineers and Computer Scientists 2009 Vol IIMECS 2009, March 18 - 20, 2009, Hong Kong.

[12] Muskan Kukreja, Stephen Albert Johnston and Phillip Stafford "Comparative study of classification algorithms for immunosignaturing data" Kukreja et al. BMC Bioinformatics 2012, 13:139 http://www.biomedcentral.com/1471-2105/13/139

[13] Mahendra Tiwari, Manu Bhai Jha, OmPrakash Yadav "Performance analysis of Data Mining algorithms in Weka" IOSR Journal of Computer Engineering (IOSRJCE) ISSN: 2278- 0661, ISBN: 2278-8727 Volume 6, Issue 3 (Sep-Oct. 2012), PP 32-41 www.iosrjournals.org

Bharat S. Makhija, is an M.E pursuing student in Computer Science and Engineering branch at H.V.P.M College of Engineering, Amravati.

Dr. Anjali B. Raut, is a Ph.D holder in data mining and fuzzy logic, currently she is Head of computer science and engineering department at H.V.P.M College of Engineering, Amravati. She has twenty years of teaching experience.