International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

Study of Classification Algorithm in Data Mining

R. Savundhariyalachmi[1], N. Pandimeena[2], P. Ramya[3]

[1, 2]M.Sc (CS & IT), Department of CS & IT, Nadar Saraswathi College of Arts and Science, Theni, Tamilnadu, India

[3]Assistant Professor, Department of CS & IT, Nadar Saraswathi College of Arts and Science, Theni, Tamilnadu, India

Abstract: ***Classification is a model finding process that is used for portioning the data into different classes according to some constraints .In other words we can say that classification is process of generalizing the data according to different instances. In this paper we present the basic classification technique. Several major kind of classification method including K-nearest neighbor classifier, Naïve Bayes, Apriori Algorithm, Decision tree induction, Support vector machine[SVM].***

Keywords: Apriori, Decision tree induction, Naïve bayes, Support Vector Machine

1. Introduction

Data mining or knowledge discovery is needed to make sense and use of data knowledge discovery in data is the non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data. Data mining consists of more than collection and managing data. It also includes analysis and prediction. They also include statistical models, mathematical algorithm and machine learning methods. people are often do mistakes while analyzing or possibly, when trying to establish relationships between multiple features. This makes it difficult for them to find solutions to certain problems. Machine learning can often be successfully applied to these problems, improving the efficiency of systems and the designs of machines. There are several application of data mining (ML) the significant of which is data mining. In particular this work is concerned with classification problems in which the output of instances admits only discrete unordered values.

2. The Apriori Algorithm

The apriori algorithm was proposed by Agrawal and Srikant in 1994.Apriori is designed to operate on database containing transaction. Apriori uses a "bottomup" approach, where frequent subsets are extended one item at a time (a step known as *candidate generation*), and groups of candidates are tested aging to the data. The algorithm terminates when no further successful extension are found. Apriori uses Breadth-First search and a Hast tree Structure to count candidate item set of length then it prunes the candidates which have a infrequent sub pattern. According to the downward closure which have an infrequent item set among candidates. Apriori is an influential algorithm for mining frequent itemsets for Boolean association rules. The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset properties, as we shall see below. Apriori is a seminal algorithm for finding frequent itemsets using candidate generation. It is characterized as a level-wise complete search algorithm using anti-monotonicity of itemsets, “if an itemset is not frequent, any of its superset is never frequent”. By convention, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order. Let the set of frequent itemsets of size k be Fk and their candidates be Ck. Apriori first scans the database and searches for frequent itemsets of size 1 by accumulating the count for each item and collecting those that satisfy the minimum support requirement. It then iterates on the following three steps and extracts all the frequent itemsets.

1) Generate Ck+1, candidates of frequent itemsets of size k +1, from the frequent itemsets of size k.
2) Scan the database and calculate the support of each candidate of frequent itemsets.
3) Add those itemsets that satisfies the minimum support requirement to Fk+1

3. Decision Tree Induction

Decision trees are trees that classify instances by sorting them based on feature values. Each node in a decision tree represents a feature in an instance to be classified, and each branch represents a value that the node can assume. Instances are classified starting at the root node and sorted based on their feature values.. A decision tree is a tree in which each branch node represents a choice between a number of alternatives, and each leaf node represents a decision. Decision tree are commonly used for gaining information for the purpose of decision -making. Decision tree starts with a root node on which it is for users to take actions. From this node, users split each node recursively according to decision tree learning algorithm. The final result is a decision tree in which each branch represents a possible scenario of decision and its outcome.

A decision tree is a Fow-chart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and leaf nodes represent classes or class distributions. The topmost node in a tree is the root node.
1. create a node N;
2. if samples 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 most common class in samples;
6. select test-attribute, the attribute among attribute-listwith the highest information gain;
7. label node N with test-attribute;
8. for each known value ai of test-attribute
9. grow a branch from node N for the condition test-attribute= ai;
10. let si be the set of samples for which test-attribute= ai;
11. if si is empty then
12. attach a leaf labeled with the most common class in samples;
13. else attach the node returned by Generate_decision_tree(si, attribute-list_test-attribute)

Decision trees can be significantly more complex representation for some concepts due to the replication problem. Solution is using an algorithm to implement complex features at nodes in order to avoid replication.

4. K-Nearest Neighbor classifier:

Nearest neighbor classifiers are based on learning by analogy. The training samples are described by n dimensional numeric attributes. Each sample represents a point in an n dimensional space. In this way, all of the training samples are stored in an n-dimensional pattern space.

When given an unknown sample, a k-nearest neighbor classifier searches the pattern space for the k training samples that are closest to the unknown sample. "Closeness" is defined in terms of Euclidean distance, where the Euclidean distance, where the Euclidean distance between two points, X=(x1, x2, ……, xn) and Y=(y1, y2, …., yn) is d(X, Y)= 2

$$d(x,y) = \sqrt{\sum(xi - yi)2}$$

classification process. One of the most straight forward instance-based learning algorithms is the *nearest neighbor* algorithm. Aha (1997) and De Mantaras and Armengol(1998) presented a review of instance-based learning classifiers. Thus, in this study, apart from a brief description of the *nearest neighbor* algorithm, we will refer to some more recent works. *k-Nearest Neighbor* (kNN) is based on the principle that the instances within a dataset will generally exist in close proximity to other instances that have similar properties (Cover and Hart, 1967). If the instances are tagged with a classification label, then the value of the label of an unclassified instance can be determined by observing the class of its nearest neighbors. The Knn locates the *k* nearest instances to the query instance and determines its class by identifying the single most frequent class label. A pseudo-code example for the instance base learning methods is illustrated.

procedure InstanceBaseLearner(Testing Instances)
for each testing instance
{
find the k most nearest instances of the training set according to a distance metric
Resulting Class= most frequent class label of the k nearest instances
}

A drawback of the basic "majority voting" classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the *k* nearest neighbors due to their large number. One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its *k* nearest neighbors. The class (or value, in regression problems) of each of the *k* nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point. Another way to overcome skew is by abstraction in data representation. Suppose that an object is sampled with a set of different attributes, but the group to which the object belongs is unknown. Assuming its group can be determined from its attributes; different algorithms can be used to automate the classification process. A nearest neighbor classifier is a technique for classifying elements based on the classification of the elements in the training set that are most similar to the test example. With the k-nearest neighbor technique, this is done by evaluating the k number of closest neighbors [8]

In pseudocode, k-nearest neighbor classification algorithm can be expressed fairly compactly [8]:
k - number of nearest neighbors
for each object *X* in the test set **do**
calculate the distance D(*X*, *Y*) between *X* and every object *Y* in the training set
neighborhood - the *k* neighbors in the training set closest to *X*
X.class SelectClass(*neighborhood*)
end for

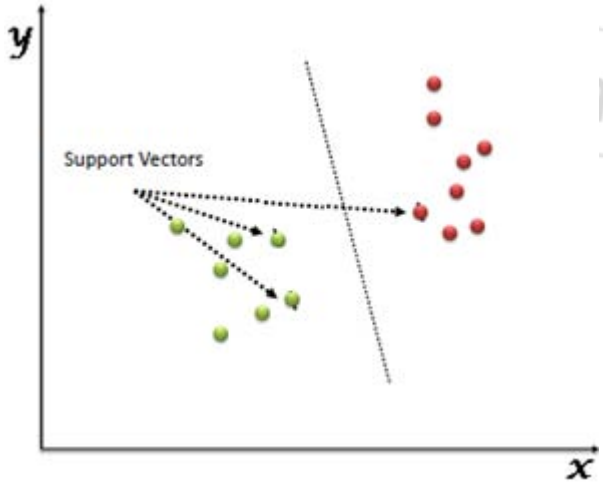
Naïve Bayes:
Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlation between the color, roundness, and diameter features. For some types of probability models, naive Bayes classifiers can be trained very efficiency in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesien Probility or using on Bayeisen methods. Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In

2004, an analysis of the Bayesian classification problem show that are sound theoretical reason for the apparently implausible efficacy of naive Bayes classifiers Given a set of objects, each of which belongs to a known class, and each of which has a known vector of variables, our aim is to construct a rule which will allow us to assign future objects to a class, given only the vectors of variables describing the future objects. Problems of this kind, called problems of supervised classification, are ubiquitous, and many methods for constructing such rules have been developed. One very important one is the naive

Bayes method—also called Bayes, simple Bayes, and independence Bayes. This method is important for several reasons. It is very easy to construct, not needing any complicated iterative parameter estimation schemes. This means it may be readily applied to huge data sets. It is easy to interpret, so users unskilled in classifier technology can understand why it is making the classification it makes. And finally, it often does surprisingly well: it may not Probabilistic approaches to classification typically involve modeling the conditional probability distribution $P(C|D)$, where C ranges over classes and D over descriptions, in some language, of objects to be classified. Given a description d of a particular object, we assign the class $argmaxcP(C = c|D = d)$. A Bayesian approach splits this posterior distribution into a prior distribution $P(C)$ and a likelihood $P(D|C)$:

5. Support vector machine

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for either classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot).



Support vector machines are supervised learning models with Associated learning algorithms that analyze data used for classification and regression analysis. An SVM model is a representation of the examples as point in space, mapped so that examples of the separate categories are divided by a clear gap that is as a wide as possible.

6. Conclusion

In the classification algorithm is described in this paper. The classification methods are typically strong in modeling interaction. The goal of classification result integration algorithms is to generate more certain precise and accurate system results.

References

[1] Xindong wu et.al,"top 10 algorithms of data Mining", springer-Verlag London, 2007.
[2] Pang –ning tan, Michael Steinbach and vipin kumar. Introduction to data Mining Addision Wesley, 2006
[3] Plilip s.yu, zhi-hua zhou, david j.hand and dan steinberg Top 10 Algorithm in data mining vol 14 no1 pp.1-37 Dec 2007
[4] T-cover and p.hart 'nearest neighbor pattern classification" IEEE Trans inf theor vol 13.nol pp 21-27 sep 2006