COMPARATIVE STUDY OF CLASSIFICATION ALGORITHMS

M. Govindarajan
Lecturer,
Dept. of CSE
Annamalai University
govindarajan_aucse@yahoo.com

RM. Chandrasekaran
Reader,
Dept. of CSE
Annamalai University
aurmc@sify.com

Prof B. Palaniappan,
DEAN – FEAT,
Annamalai University
head_aucse@yahoo.com

Abstract

The problem of classification of records in a mushroom like database is a very difficult problem and tedious process. Study reveals that classifying the records discloses some useful patterns for selective marketing, financial forecast, and many other applications, hence it has attracted a lot of attention in recent data mining research. One of the important problems in data mining is the Classification-rule learning that involves finding rules that partition given data into predefined classes. In the data-mining domain where millions of records and a large number of attributes are involved, the execution time of existing algorithms can be exhaustive.

Therefore, efficient classification algorithms in mushroom database have to be examined meticulously. A comparative study of Classification algorithms like K-Nearest Neighbors, Naïve Bayes, BVDecompose, C4.5, AdaBoostM1 using mushroom database based on time complexity was done and the best algorithm is identified. Existing algorithm is modified to implement the multithreading concept and resulting in a better algorithm called as "AdaBoostMulti". It is observed that "AdaBoostMulti" outperforms the earlier algorithm.

Keywords*: Data mining, classification, comparative study and time Complexity.*

1. Introduction

Data mining refers to extracting or "mining" knowledge from large amounts of data. There are many other terms carrying a similar or slightly different meaning to data mining, such as knowledge mining from databases, knowledge extraction, data/pattern analysis, data archeology, and data dredging.

Classification

By simple definition, classification deals with analyzing a set of data and generates a set of grouping rules which can be used to classify future data. For example, one may classify diseases and provide the symptoms which describe each class or subclass. This has much in common with traditional work in statistics and machine learning. However, there are important new issues which arise because of the sheer size of the data.

2. Classification Algorithms

In Data classification one develops a description or model for each class in a database, based on the features present in a set of class-labeled training data. There have been many data classification methods studied, including decision-tree methods, such as C4.5, statistical methods, neural networks, rough sets, database-oriented methods etc.

2.1 K Nearest Neighbors

One common classification scheme based on the use of distance measures is that of the K nearest neighbors [3] (KNN). The KNN technique assumes that the entire training set includes not only the data in the set but also the desired classification for each item. In effect, the training data in the set but also the desired classification for each item. In effect, the training data become the model. When a classification is to be made for a new item, its distance to each item in the training set must be determined. Only the K closest entries in the training set are considered further. The new item is then placed in the class that contains the most items from this set of K closest items. If there q elements in the training set, this is O(q). Given n elements to be classified, this becomes O(nq) problem. Given that the training data are of a constant size, this can be then be viewed as an O(n) problem.

```
Algorithm
Input :
        T // Training data
        K  // Number of neighbors
        t    // Input tuple to classify
Output:
        c   // class to which t is assigned
             KNN Algorithm
        N = 0;
```

```
        // Find set of neighbors, N for t
For each d ε T do
   if | N | ≤ K, then
      N = N U {d}
    else
      if ∃u ε N such that
        sim(t,u) ≤ sim (t,d) then
         begin
            N = N – {u};
            N = N U {d};
         end
// Find class for classification
c = class to which the most u ε N are classified;
```

2.2 Naïve Bayes

Assuming that the contribution by all attributes are independent and that each contributes equally to the classification problem, a simple classification scheme called Naïve Bayes [3] classification has been proposed that is based on Bayes rule of conditional probability. By analyzing the contribution of each "independent" attribute, a conditional probability is determined. A classification is made by combining the impact that the different attributes have on the prediction to be made. The approach is called "naïve" because it assumes the independence between the various attribute values. Given a data value x_i the probability that a related tuple t_i, is in class C_j is described by $P(C_j|\ t_i)$. Bayes theorem allows us to estimate the posterior probability $P(C_j|x_i)$ and then $P(C_j|t_i)$. Given a training set, the naïve Bayes algorithm first estimates the prior probability $P(C_j)$ for each class by counting how often each class occurs in the training data. For each attribute x_i, the number of occurrences of each attribute value x_i can be counted to determine $P(x_i)$. Similarly, the probability $P(x_i|C_j)$ can be estimated by counting how often each value occurs in the class in the training data. When classifying a target tuple, the conditional and prior probabilities generated from the training set are used to make the prediction. This is done by combining the effects of the different attribute values from the tuple. Suppose that tuple t_i has p independent attribute values $\{x_{i1},x_{i2},\ldots..x_{ip}\}$. From the descriptive phase, it is known that $P(x_{ik}|C_j)$, for each class C_j and attribute $x_{ik.}$ Then $P\ (t_i \mid C_j)$ is estimated by

$$P(t_i|C_j) = \prod_{k=1}^{p} P(x_{ik}|C_j)$$

$P(C_j)$ for each class and the conditional probability $P\ (t_i|C_j)$. $P(t_i)$, the likelihood that t_i is in each class can now be estimated. The probability that t_i is in a class is the product of the conditional probabilities for each attribute values. The posterior probability $P\ (C_j|t_i)$, is then found for each class. The class with the highest probability is the one chosen for the tuple.

2.3 Bias Variance Decomposes (BVDecompose)

The Bias Variance Decomposition [5] of the expected cost into its components and geometric views of this decomposition, in particular relating it to quadratic loss in Euclidean spaces is now considered.

The Decomposition

The general result involving the expected zero-one loss is represented as, E(C), where the (implicit) conditioning event is arbitrary. The standard conditioning used in conventional sampling theory statistics is: a single test point, target, and training set size.

$$E(C) = 1 - \sum_{y\varepsilon Y} P(Y_H=Y_F=y)$$
$$= \sum_{y\varepsilon Y} -P(Y_H=Y_F=y) + \sum_{y\varepsilon Y} P(Y_H=y)\,P(Y_F=y)$$
$$+ \sum_{y\varepsilon Y} [-P(Y_H=y)\,P(Y_F=y) + \tfrac{1}{2} P(Y_F=y)^2 + \tfrac{1}{2} P(Y_H=y)^2]$$
$$+ \quad [\tfrac{1}{2} - \tfrac{1}{2} \sum_{y\varepsilon Y} P(Y_H=y)^2] + [\tfrac{1}{2} - \tfrac{1}{2} \sum_{y\varepsilon Y} P(Y_F=y)^2]$$

Rearranging the terms

$$E(C) = \sum_{y\varepsilon Y} [P(Y_H=y)\,P(Y_F=y) - P(Y_F=Y_H=y)] + \text{("covariance")}$$

$$\tfrac{1}{2} \sum_{y\varepsilon Y} (P\ (Y_F=y) - P(Y_H=y))^2 + \text{("bias}^2\text{")}$$

$$\tfrac{1}{2}\Big(1 - \sum_{y\varepsilon Y} P(Y_H=y)^2\Big) + \text{("variance")}$$

$$\tfrac{1}{2}\Big(1 - \sum_{y\varepsilon Y} P(Y_F=y)^2\Big) \quad \text{("}\sigma^2\text{")}$$

The main interest is E(C | f, m), the expected cost. One way to evaluate this quantity is to write it as

$$\sum_x P(x)\, E(C \mid f, m, x).$$

Y_H and Y_F are independent when one conditions on f and x, hence the ``covariance" term vanishes. So

$E(C)=\Sigma_x P(x) \quad (\sigma^2{}_x+bias^2{}_x+ variance_x)$

$P(Y_F = y \mid x)$ is the probability (after any noise is taken into account) that the fixed target takes on the value y at point x. To understand the quantity $P(Y_H=y|x)$, one must write in full as

$P(Y_H=y \mid f,m,x)$

$= \Sigma_d P(d \mid f,m,x) P(Y_H=y|d,f,m,x)$

$= \Sigma_d P(d|f,m)P(Y_H=y \mid d,x)$

In this expression, $P(d \mid f,m)$ is the probability of generating training set d from the target f, and $P(Y_H=y \mid d,x)$ is the probability that the learning algorithm makes guess y for point x in response to the training set d. so $P(Y_H=y \mid x)$ is the average Y value guessed by the learning algorithm for point x.

2.4 C4.5 algorithm

The objective of decision tree classification is to iteratively partition the given data set in subsets where all elements in each final subset belong to the same class. The concept used to quantify information is called entropy. Entropy is used to measure the amount of uncertainty. In the divide and conquer approach such as that used with decision trees, the division results in multiple probabilities whose sum is 1. To measure the information associated with this division, we must be able to combine the information associated with both events. This can be performed by adding the two values together and taking into account the probability that each occurs. The C4.5 [3] takes into account cardinality of each division into account for splitting. This approach uses the Gain Ratio as opposed to Gain. The GainRatio is defined as

$Gain\ (D,S) = Gain(D,S)/(H(|D_1|/|D| \ldots |D_s|/|D|))$

There are primarily two pruning strategies used in C4.5

- With subtree replacement, a subtree is replaced be a leaf node if this replacement results in an error rate close to that of the original tree. Subtree replacement works from the bottom of the tree up to the root.
- Another pruning strategy, called subtree raising, replaces a subtree by its most used subtree. Here a subtree is raised from its current location to a node higher up in the tree. The increase in error rate for this replacement must also be determined.

The algorithm considers all the possible tests that can split the data set and selects a test that gives the best information gain. For each discrete attribute, one test with outcomes as many as the number of distinct values of the attribute is considered. For each continuous attribute, binary tests involving every distinct values of the attribute are considered.

2.5 AdaBoostM1

Boosting is a way of combining the performance of many “weak” classifiers to produce a powerful “committee”. Boosting was proposed in the Computational Learning Theory literature and has since received much attention. The AdaBoost [2] procedure trains the classifiers $f_m(x)$ on weighted versions of the training sample, giving higher weight to cases that are currently misclassified. This is done for a sequence of weighted samples, and then the final classifier is defined to be a linear combination of the classifiers from each stage. Let L be the chosen “weak” learning algorithm and T be the number of iterations to perform.

Algorithm AdaBoost.M1(Instance set, L)
  For i ← 1 to | Instance set |
    D_1 (i) <- 1/ |instance set|
  For t = 1 to T
    h_t <- the model induced by L from
      Instance set with distribution D_t
    ε_t <- $\sum_{i:\ ht(xi) \neq yi} D_t(i)$
    if $\varepsilon_t > 0.5$
      T <- t – 1
      Abort loop
    β_t <- $\varepsilon_t / (1 - \varepsilon_t)$
    For i <- 1 to | Instance set |
      $D_{t+1}(i)$ <- $D_t(i) / Z_t$ x $\begin{cases} \beta_t \text{ ; if } h_t(x_i) = y_i \\ 1 \text{ ; otherwise} \end{cases}$

where Z_t is a normalisation constant,chosen so that D_{t+1} will be a distribution

$h_{final}(x) <- \text{argmax}_{y \varepsilon Y} \sum_{t: ht(x)=y} \log(1/\beta_t)$

2.6 AdaBoostMulti

AdaBoostMulti is a modification of the existing AdaBoostM1 algorithm. In the implementation of the AdaBoostM1, the "weak" classifiers are run sequentially. In AdaBoostMulti, the independent "weak" classifiers are run concurrently, thus increasing the speed of execution, i.e. decreasing the overall execution time.

3. Performance Evaluation

The performance of classification algorithms using mushroom database was done. Details of the experiments including performance graphs and detailed explanation of the results are as follows:

Table 1. Time complexity

Algorithm	Time (Sec)
K-Nearest-Neighbor (KNN)	333.7
Naïve Bayes	2
BVDecompose	11.5
C4.5	14.8
AdaBoostM1	32.8
AdaBoostMulti	9.7

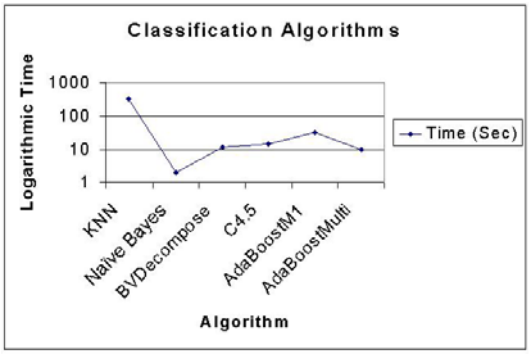

Figure 1

Test Case

Comparative study of the performance of classification algorithms using mushroom database was done.

Database size: 8124 records

Features

All the characteristics involved in uniquely identifying a mushroom species like cap color, cap surface, veil color, veil type, ring number etc.

Individuality

- Easy to obtain
- Contains diverse data

4. Conclusion

Based on the performance chart of various algorithms examined, it has been concluded that the Naïve Bayes algorithm outperforms the other algorithms in classification in time constraints, and the AdaBoostMulti outperforms the existing AdaBoostM1 algorithm in time constraint by a factor of 3:1.

5. Acknowledgement

Authors gratefully acknowledge the authorities of Annamalai University for the facilities offered and encouragement to carry out this work

6. Reference

[1] Jiawei Han and Micheline Kamber, "*Data Mining Concepts and Techniques*"

[2] Jerome Friedman, Trevor Hastie, Robert Tibshirani: "*Additive Logistic Regression:a Statistical View of Boosting*" (1999)

[3]. Margaret H.Dunham, "*Data Mining – Introductory and Advanced Topics*" (pages 164-173)

[4] J. Han, J. Chiang, S. Chee, J. Chen, Q. Chen, S. Cheng, W. Gong, M. Kamber, K. Koperski, G. Liu, Y. Lu, N. Stefanovic, L. Winstone, B. Xia, O. R. Zaiane, S. Zhang, H. Zhu, `*DBMiner: A System for Data Mining in Relational Databases and Data Warehouses''*, Proc. CASCON'97: Meeting of Minds, Toronto, Canada, November 1997.

[5] Ron Kohavi, David H.Wolpert, "Bias Plus Variance Decomposition for Zero-One Loss Functions" (1996)