Classification Machine Problem

Prakyath Jaladhar
jaladha2@illinois.edu

This is a report for the CS412 Assignment 4 - Classification MP. In this report I will be explaining the two algorithm used to classify the given data into two classes. Both the algorithms have been implemented in C++. The code in the 'code' folder consists of 4 files:

1. naivebayes.h and naivebayes.cpp
2. adaboost.h and adaboost.cpp
3. Makefile

To build the 'NaiveBayes' and 'NBAdaBoost' application please run 'make' command. After this step you should be able to run both applications using the following commands.

./NaiveBayes training_file test_file
./NBAdaBoost training_file test_file

Step 1: Introduction of the classification methods used

1. Naive Bayes :

Naïve Bayes Classifier is based on the Bayes' Theorem. According to the theorem,

$$P(H|X) = P(X|H) \times P(H) / P(X)$$

where X is a data tuple and H is the hypothesis that X belongs to a specified class $C_{i,}$ and then we compare the posteriori probability of each hypothesis H. By derivation and simplification, we compare, train and classify data tuple based on the magnitude of $P(X|C_i)P(C_i)$ for various C_i , i = 1...k. Information about $P(C_i)$ is learned from the labels of training set. The class conditional independence is assumed, which greatly simplifies the calculation of $P(X|C_i)$:

$$P(X|C_i) = \prod_{k=1}^{n} P(X_k | C_i)$$
$$= P(X_1 | C_i) \times P(X_2 | C_i) \times \cdots \times P(X_n | C_i).$$

To predict the class label of X, $P(X|C_i)P(C_i)$ is evaluated for each class C_i . The classifier predicts that the class label of tuple X is the class, if and only if

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

In this assignment , we only focus on binary class classification (-1/+1). In addition, *Laplacian correction technique* is implemented in order to tackle the zero probability problem. As mentioned in the problem statement, "for a data instance (one line), if the value of an attribute is 0, this attribute is omitted in this line. It does not mean this attribute is missing. For all dataset, you can use the largest index in both training and test file as the number of attributes." largest index was found in both the training as well as testing files. To counter the problem of zero probability, I have implemented

Laplacian correction technique, in which while finding out the probability of an attribute, if the value is equal to zero, I add 1 to the numerator and add the maximum index value to the denominator.

The implementation details is as follows:

1. Both the training and test files are scanned to get the maximum index value and is stored.
2. Both the training and test files are read into the system and the number of attributes is stored.
3. I have implemented the train function which then trains the naïve bayes classifier, all the probabilities calculated are subjected to Lapacian correction and are stored in a 3-dimensional probability table.
4. Then I call the "predict" function which calculates the posteriori probabilities of all the tuples of the test data and compares both and classifies the tuple and stores the result in a vector.
5. The last function is "calculate measures" checks the given training and test class labels and calculates the 4 measures each for both the training and testing file.
    1. True Positive
    2. False Negative
    3. False Positive
    4. True Negative

2. AdaBoost:

The AdaBoost classifier uses the same functions used by the naïve bayes classification method. There are 2 new functions which are the "adaBoostTrain" and the "adaBoostPredict".

1. In adaBoostTrain I have a loop to iterate in order to build k = 6 weak classifiers. In each I did sampling according to the weight (at first the weight is equal and sum = 1) of tuples in the input training dataset and then use the train function to train a weak classification model. I have used the *seed function* in order to make sure that there are different random numbers generated and different tuples are considered as samples.

Also I calculated the error here according to the formula,

$$\boldsymbol{Error(M_j) = \sum_{k=1}^{j} w_j \times err(X_j)}$$

in which sums the weight of misclassified X_j within the d tuples of the sample. If the error rate is > 0.5 , this classifier is considered weak and is discarded and the whole steps will be repeated again until a classifier gives an error of < 0.5. Then the weights of tuples *correctly predicted* are all calculated before normalizing with all other weights of tuples together using the formula,

$$\boldsymbol{New\_weight = Old\_weight \times Error(M_j) / 1 - Error(M_j)}$$

"adaBoostPredict" is similar to the "predict" function, which picks each tuple in the testing data and calculates the posteriori probabilities of the tuple for the two classes and classifies them. This is done for each k = 6 classifiers. First I initialized the class-weight to 0 for both positive and negative class. Then there is a loop for k = 6 weak classifiers. Within each loop, I find out the weight of classifier's vote using the formula,

$$\boldsymbol{W_i = log\ (1 - error(M_i) / error(M_i))}$$

I then add it to the predicted class label for the input tuple according to the k-th weak classifier. Finally the tuple is classified using the largest class weight.

STEP 2: Model Evaluation Measures.

I have made several runs of the Adaboost algorithm in-order to obtain the best result. I have pasted the results of all the data sets in this section for both the algorithms.

1. *Adult dataset:*
a) ./NaiveBayes adult.train adult.test

i. adult.train :

True Positive: 323 False Negative: 72 False Positive: 252 True Negative: 958
Accuracy: 0.798131 Error Rate: 0.201869
Sensitivity: 0.817722 Specificity: 0.791736
Precision: 0.561739 F-1 Score: 0.665979
F-0.5 Score: 0.599258 F-2 Score: 0.74942

ii. adult.test:

True Positive: 5953 False Negative: 1493 False Positive: 4928 True Negative: 18582
Accuracy: 0.792577 Error Rate: 0.207423
Sensitivity: 0.79949 Specificity: 0.790387
Precision: 0.5471 F-1 Score: 0.649643
F-0.5 Score: 0.583971 F-2 Score: 0.731956

b) ./NBAdaBoost adult.train adult.test

i. adult.train :

True Positive: 301 False Negative: 94 False Positive: 219 True Negative: 991
Accuracy: 0.804984 Error Rate: 0.195016
Sensitivity: 0.762025 Specificity: 0.819008
Precision: 0.578846 F-1 Score: 0.657923
F-0.5 Score: 0.608081 F-2 Score: 0.716667

ii. adult.test:

True Positive: 5604 False Negative: 1842 False Positive: 4051 True Negative: 19459
Accuracy: 0.809633 Error Rate: 0.190367
Sensitivity: 0.752619 Specificity: 0.82769
Precision: 0.580425 F-1 Score: 0.6554
F-0.5 Score: 0.608258 F-2 Score: 0.710464

2. Breast cancer dataset:
a) ./NaiveBayes breast_cancer.train breast_cancer.test

i. breast_cancer.train :

True Positive: 30 False Negative: 26 False Positive: 20 True Negative: 104
Accuracy: 0.744444 Error Rate: 0.255556
Sensitivity: 0.535714 Specificity: 0.83871
Precision: 0.6 F-1 Score: 0.566038
F-0.5 Score: 0.585937 F-2 Score: 0.547445

ii. breast_cancer.test :

True Positive: 15 False Negative: 14 False Positive: 14 True Negative: 63
Accuracy: 0.735849 Error Rate: 0.264151
Sensitivity: 0.517241 Specificity: 0.818182
Precision: 0.517241 F-1 Score: 0.517241
F-0.5 Score: 0.517241 F-2 Score: 0.517241

b) ./NBAdaBoost breast_cancer.train breast_cancer.test

i. breast_cancer.train :

True Positive:35 False Negative:21 False Positive:19 True Negative:105
Accuracy: 0.777778 Error Rate: 0.222222
Sensitivity: 0.625 Specificity: 0.846774
Precision: 0.648148 F-1 Score: 0.636364
F-0.5 Score: 0.643382 F-2 Score: 0.629496

ii. breast_cancer.test :

True Positive:13 False Negative:16 False Positive:16 True Negative:61
Accuracy: 0.698113 Error Rate: 0.301887
Sensitivity: 0.448276 Specificity: 0.792208
Precision: 0.448276 F-1 Score: 0.448276
F-0.5 Score: 0.448276 F-2 Score: 0.448276

3. Led dataset:

a) ./NaiveBayes led.train led.test

i. led.train :

True Positive:399 False Negative: 239 False Positive: 92 True Negative: 1357
Accuracy: 0.841399 Error Rate: 0.158601
Sensitivity: 0.625392 Specificity: 0.936508
Precision: 0.812627 F-1 Score: 0.70682
F-0.5 Score: 0.766718 F-2 Score: 0.655603

ii. led.test :

True Positive:207 False Negative: 144 False Positive: 41 True Negative: 742
Accuracy: 0.836861 Error Rate: 0.163139
Sensitivity: 0.589744 Specificity: 0.947637
Precision: 0.834677 F-1 Score: 0.691152
F-0.5 Score: 0.770663 F-2 Score: 0.626513

b) ./ NBAdaBoost led.train led.test

i. led.train :

True Positive:426 False Negative: 212 False Positive: 112 True Negative: 1337
Accuracy: 0.844753 Error Rate: 0.155247
Sensitivity: 0.667712 Specificity: 0.922705
Precision: 0.791822 F-1 Score: 0.72449
F-0.5 Score: 0.763441 F-2 Score: 0.68932

ii. led.test :

True Positive:235 False Negative: 116 False Positive: 50 True Negative: 733
Accuracy: 0.853616 Error Rate: 0.146384
Sensitivity: 0.669516 Specificity: 0.936143
Precision: 0.824561 F-1 Score: 0.738994
F-0.5 Score: 0.788062 F-2 Score: 0.695678

4. Poker dataset:

a) ./NaiveBayes poker.train poker.test

i. poker.train :

True Positive:740 False Negative: 7 False Positive: 284 True Negative: 10
Accuracy: 0.720461 Error Rate: 0.279539
Sensitivity: 0.990629 Specificity: 0.0340136
Precision: 0.722656 F-1 Score: 0.835686
F-0.5 Score: 0.763989 F-2 Score: 0.922233

ii. poker.test :

True Positive:448 False Negative: 11 False Positive: 217 True Negative: 2
Accuracy: 0.663717 Error Rate: 0.336283
Sensitivity: 0.976035 Specificity: 0.00913242
Precision: 0.673684 F-1 Score: 0.797153
F-0.5 Score: 0.718179 F-2 Score: 0.895642

b) ./ NBAdaBoost poker.train poker.test

i. poker.train :

True Positive:684 False Negative: 63 False Positive: 245 True Negative: 49
Accuracy: 0.704131 Error Rate: 0.295869
Sensitivity: 0.915663 Specificity: 0.166667
Precision: 0.736276 F-1 Score: 0.816229
F-0.5 Score: 0.766301 F-2 Score: 0.873117

ii. poker.test :

True Positive:420 False Negative: 39 False Positive: 193 True Negative: 26
Accuracy: 0.657817 Error Rate: 0.342183
Sensitivity: 0.915033 Specificity: 0.118721
Precision: 0.685155 F-1 Score: 0.783582
F-0.5 Score: 0.721402 F-2 Score: 0.857493

<u>STEP 3: Parameters chosen during implementation</u>

As discussed in STEP 1 about the implementation details used in the program, there are a number of parameters chosen while implementing the 2 algorithms,

1. Laplacian correction: The idea is to eliminate the possibility of getting a 0 probability if feature is not seen. To counter this problem I have implemented Laplacian correction technique, in which while finding out the probability of an attribute, if the value is equal to zero, I add 1 to the numerator and add the maximum index value to the denominator.
2. Number of weak classifiers k = 6 for AdaBoost algorithm. Choosing the value of k depends on how algorithm is implemented. There is not optimal solution for this problem, I have chosen k = 6 as I found the result is better compared to Naive Bayes and increase k to be greater than 6 or decreasing it didn't have much influence on improving the result.

<u>STEP 4: Conclusion - Whether the ensemble method improves the performance of the basic classification method</u>

For most of the cases in these four dataset. AdaBoost performs better than pure Naive Bayes classification. Because the AdaBoost algorithm does sampling and voting, making the weight of each tuples different, which makes it possible for the classifier to pay more attention to those misclassified tuples.