ICS 171
Fall 2005

Project
(due Dec8 2005)

Learning

In this project you will implement one learning algorithms studied in class. Two datasets will be provided to train/test the algorithms.

1. Naive Bayes classifier

For a bonus you can implement the boosted Naive Bayes algorithm.

Toy Dataset

The first dataset is a toy problem and is given below. Each instance has 5 discrete attributes (Outlook, Temperature, Humidity, Windy, Play Golf). The last attribute is the class.

The first 10 instances will be your training set. The remaining four instances are your test set. The performance of the algorithms is measured by accuracy, i.e. the ration between the number of instances classified correctly and the total number of test instances.

No.	Outlook	Temperature	Humidity	Windy	Play Golf?
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcast	hot	high	false	Y
4	rain	mild	high	false	Y
5	rain	cool	normal	false	Y
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	Y
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	Y
10	rain	mild	normal	false	Y
11	sunny	mild	normal	true	Y
12	overcast	mild	high	true	Y
13	overcast	hot	normal	false	Y
14	rain	mild	high	true	N

Real World Dataset

ICS 171
Fall 2005

A second dataset is a real-world problem instance. You can download it from the class website (http://www.ics.uci.edu/~welling/teaching/ICS171Fall05/Xtrn.txt, and http://www.ics.uci.edu/~welling/teaching/ICS171Fall05/Ytrn.txt).
The first file contains the training attributes: a 54x500 matrix with 0's and 1's. The second file a 1x500 matrix with the labels (1 for spam, 0 for non-spam).
Each attribute indicates the presence or absence of certain words in an email. Your task is to build your own "spam-assassin" that can filter out future spam messages. On this dataset, a simple naive Bayes classifier achieves around 39% training error. If you don't get that, you may have a bug in your code.

Naive Bayes Classifier (required)

The naive Bayes classifier fits a very simple model to each class separately, and uses Bayes rule to make the classification decision. Report the classification performance on the training set.

Boosted Naive Bayes Classifier (BONUS)

The boosted version uses the code you already have for the naive Bayes classifier and applies it iteratively on re-weighted data. You should determine some value for the number of boosted rounds (any number is OK, but some clearly do better than others). Show plots of training error and testing error as a function of the boosting rounds. I got about 37% training error using this method, so beware that the improvement may not be very large.

Extra Credit (double BONUS).

For extra-credit, you'll have to implement a cross-validation procedure to figure out the number of rounds to boost. The winner on the held-out test set wins the competition. Other ways to get extra bonus are to implement a number of other classifiers treated in class and comparing performance (e.g. k-nearest neighbors, logistic regression, decision stump boosting).

What to hand in

- Source code must be submitted using the class drop-box in EEE.
- You will hand in a report containing the description of your method as well as examples demonstrating that your program works correctly. Limit your discussion to 2 pages.

General Instructions

- You may discuss the problem with other people but must develop your program independently.
- Your program will have four arguments:

- args[0] – the algorithm (e.g. 0 for NB, 1 for BNB, etc).
- args[1] - input file containing the training dataset.
- args[2] – input file containing the test dataset.
- args[3] – output file with results.

- The training/test files are formatted as follows. Each line corresponds to a single observation. Attribute values are comma separated (the first two observations in the toy problem are given below.

```
sunny,hot,high,false,N
sunny,hot,high,true,N
```

- The results file should be formatted in a similar manner and it should contain the test instances together with the class (the one computed by your classifier). You should also report the accuracy of your classifier.
- You are **required** to use C/C++, Java or MATLAB programming languages.
- Project descriptions will be discussed in sections. Feel free to contact your TA if you have any uncertainties.