Naive Bayes Classifier

Johnson Lin
Yongchen@ucsb.edu
Perm:8485336

Architecture

For machine problem 1, I used Python to create the Naive Bayes Classifier. The program starts by reading in the name of the training data file through the command line and then proceeds to read through the entire file, placing the data of each instance into a corresponding array. The program then proceeds to calculate certain statistics relating to different attributes (discrete attributes are handled differently). These statistics can then be used to calculate the conditional probability of an attribute given a classification (high or low performance) which can be used to classify any individual instance. Finally, the program will read through the testing data file (name read from the command line) and use the calculated probabilities to predict a label and output it. The program also includes code that calculates the accuracy of the classifier given a specific set of testing and training data as well as code that prints out training and classification times.

Preprocessing

Each instance has its data/attributes placed in a different array. For each instance in the training data, we will first look at the label and then append each attribute to a corresponding array depending on whether the instance is a high or low performer. These arrays will later be used to calculate statistics relating to each attribute. The program also uses several variables to keep track of the number of males and females as well as the number of males and females that are high performers and the number of males and females that are low performers. These counts are updated as we parse through the training data and note the label and gender of each instance.

Model building

After the attributes of each instance are parsed into separate arrays, the classifier is trained by calculating the mean and standard deviation for each array. In other words, the mean and standard deviation for each attribute is calculated twice: once for the attributes of high performers and one for the attributes of low performers. These statistics can then be used by the classifier to calculate the conditional probability of a particular attribute given a label using the Gaussian distribution pdf. For discrete attributes (gender) we will simply use the counts to calculate the conditional probabilities.

Results

Accuracies :
Accuracy on training data = 77.15%
Accuracy on public testing data = 78.02%
Timing :
Classifier training time = 0.1926 s
Classifier testing time on public testing data = 0.0520
Classifier testing time on training data = 0.4449

Challenges

I observed that female athletes tended to score lower in areas such as height and weight compared to male athletes although a similar proportion of female athletes were high performers. Since male athletes tended to score significantly higher in these areas, the average became overinflated and would be biased against females. The solution was to have the classifier implement smoothing to avoid female athletes from being unfairly penalized. Without this, the classifier would automatically associate attributes such as a higher height and weight with greater performance and a large proportion of female athletes would be categorized inaccurately. Another challenge was figuring out how to efficiently parse the training data. Initially, I tried to directly index each line to get the attributes; however, certain attributes had different numbers of significant figures for different instances and this made indexing difficult. Luckily, Python has a useful split() function which allowed me to split each line by "," and store the line in a list. This made parsing each instance much easier.

Weaknesses

A major weakness in this method is that it assumes every attribute follows a Gaussian distribution which is likely not the case in reality. A solution to this would be to plot all instances of an attribute and try to identify which distribution more accurately captures the specific attribute. Then the classifier could use the probability density of that distribution to calculate a more accurate conditional probability. This is certainly something I would have liked to try if more time was available. I somewhat accounted for the classifier being biased against female athletes, but I did not do the same for males. Of course, female high performers would also cause certain attributes such as height and weight to have lower averages and this would cause the classifier to be biased against male athletes who have a higher height and weight on average. I would have also liked to implement smoothing to account for this.