**Assignment 3**

Zezhou Li

10405265

In this assignment, I am going to use Matlab to implement Naïve Bayes algorithm which is a very useful supervised machine learning algorithm. It is an intuitive method that uses the probabilities of each attribute belonging to each class to make a prediction.

The basic equation of the Bayes formula is shown as the following:

Naïve Bayes simplifies the calculation of probabilities by assuming that the probability of each attribute belonging to a given class value is independent of all other attributes. This is a strong assumption but results in a fast and effective method. To make a prediction we can calculate probabilities of the instance belonging to each class and select the class value with the highest probability.

First step is to import the data set, the given data set contains 2400 instances and the dimension of the data is 150 which means for each sample data, there are 150 different attributes, according to the assumption of the Naïve Bayes algorithm, these 150 attributes are independent of each other.

In Matlab, we can simply call the function ‘xlsread()’ to read in the data set.

|  |
| --- |
| x = xlsread('Data.xls');  label = xlsread('label.xls'); |

There are two files, first file contains the all of the sample data, the second file contains the corresponding labels to the sample data. We read in two files to ‘x’ and ‘label’ variables.

To train and test our Naïve Bayes algorithm, we not only need training data set, but also the test set. Here we set the ratio of the training set to test set as 7/3 since it is a proper separation.

To do so, we simply choose first 1680 sample data as the training data set and the rest of the data as test set.

|  |
| --- |
| train\_x = x(1:1680,:);  train\_label = label(1:1680,:);    test\_x = x(1681:end,:);  test\_label = label(1681:end,:); |

We can then also get the maximum and the minimum label from the label set.

|  |
| --- |
| max\_label = max(train\_label);  min\_label = min(train\_label); |

From the outputs, we can know that there are 9 classes since the maximum label is 9 and minimum label is 1. It means that we should deal with multi-class problem.

Next few steps are important since it will finally influence the results of our algorithm, these steps are mainly to summarize the data we had. The Naïve Bayes model is comprised of a summary of the data in the training dataset. This summary is then used when making predictions.

The summary of the training data collected involves the mean and the standard deviation for each attribute, by class value. These are required when making predictions to calculate the probability of specific attribute values belonging to each class value.

To summarize the data, firstly we need to separate the data by its class so that we can calculate the statistics by each class. We can do that by creating a map of each class value to a vector of instances that belong to that class.

|  |
| --- |
| n = length(train\_x);  data\_map = containers.Map('KeyType', 'int32', 'ValueType','any');  data\_map = initializeMap(data\_map);  for i = 1:n      k = train\_label(i);      data\_map(k) = [data\_map(k);train\_x(i,:)];  end |

After doing so, we will get a map variable called ‘data\_map’ which separate the instances of the training data set to each class.

After separating data into different classes, we need to calculate the mean of each attribute for a class value. The mean is the central middle or central tendency of the data, and we will use it as the middle of our Gaussian distribution when calculating probabilities.

We also need to calculate the variance of each attribute for a class value. The variance describes the variation of spread of the data, and we will use it to characterize the expected spread of each attribute in our Gaussian distribution when calculating probabilities.

|  |
| --- |
| data\_summarize = containers.Map('KeyType', 'int32', 'ValueType','any');  data\_summarize = initializeMap(data\_summarize);  for i = min\_label:max\_label      one\_class = data\_map(i);      data\_summarize(i) = [mean(one\_class);var(one\_class)];  end |

After all these steps, we get the statistical values we need to make the Naïve Bayes prediction. Thus, what we need to do then is to use these values to make predictions of our test data set.

Making predictions involves calculating the probability that a given data instance belongs to each class, then selecting the class with the largest probability as the prediction.

We can use a Gaussian function to estimate the probability of a given attribute value, given the known mean and standard deviation for the attribute estimated from the training data.

First is to calculate the Gaussian probability whose equation is shown as below:

|  |
| --- |
| function y = calculateProbability(x, mean, variance)      exponent = exp(-(x-mean)^2/(2\*variance));      y = 1 / (sqrt(2\*pi) \* sqrt(variance)) \* exponent;  end |

Next is to calculate the class probability, we can combine the probabilities of all the attribute values for a data instance and come up with a probability of the entire data instance belonging to the class. We combine probabilities together by multiplying them.

|  |
| --- |
| function y = calculateClassProb(summaries, inputVector)      probabilities = containers.Map('KeyType', 'int32', 'ValueType','any');      probabilities = initializeMap(probabilities);      for i = 1:9          probabilities(i) = 1;          temp = summaries(i);          for j = 1:size(summaries(i),2)              mean = temp(1,j);              variance = temp(2,j);              x = inputVector(j);              probabilities(i) = probabilities(i) \* calculateProbability(x,mean,variance);          end      end      y = probabilities;  end |

Finally, we can make the predictions. Now that can calculate the probability of a data instance belonging to each class value, we can look for the largest probability and return the associated class.

|  |
| --- |
| function y = predict(summaries, inputVector)      probabilities = calculateClassProb(summaries, inputVector);      bestLabel = 0;      bestProb = -1;      for i = 1:9          if bestLabel == 0 || probabilities(i) > bestProb              bestProb = probabilities(i);              bestLabel = i;          end      end      y = [bestLabel;bestProb];  end |

To predict, we need to input the entire test data to the predict function and get the best label for each instance.

|  |
| --- |
| function y=getPrediction(summaries, testSet)      predictions = [];      for i = 1:length(testSet)          result = predict(summaries, testSet(i,:));          predictions = [predictions;result(1,:)];      y = [predictions;result(2,:)];  end |

The predictions can be compared to the class values in the test\_label dataset and a classification accuracy can be calculated as an accuracy ratio between 0% and 100%.

|  |
| --- |
| correct = 0.0;  for i = 1:length(test\_x)      if test\_label(i) == predictions(i)          correct = correct + 1.0;      end  end  correct/length(test\_x) |

Till now, we have completed entire Naïve Bayes algorithm in Matlab, we can run the program to see the accuracy of this method:

Accuracy = 62.64%.

Well, the result is now as good as we predict, the reason of such poor accuracy may be the insufficient amount of data.