CSCE 478 Machine Learning Homework 3

Jakob Snyder(jsnyde) and Clayton Henderson(chenders)

October 31, 2016

1 Problem

1.1 Introduction

The Naive Bayes classifier presented an interesting challenge for our group, as it is distinctly unique from the previous algorithms that we have implemented during this class. Naive Bayes does not 'learn' like an artificial neural network or a decision tree. Instead, the Naive Bayes algorithm relies solely on a statistical probability function to predict the most likely label for a set of attributes. Our implementation of this algorithm is very straight-forward, and our results showed a surprising similarity to the findings of previous algorithms in this class. The data sets we used to test the system were: voting records, monks, and tic-tac-toe.

1.2 Implementation

Our implementation of the Naive Bayes classifier is simple. First, we parse the input data sets from voting data, monks, and tic-tac-toe. We convert all attribute data into enumerations to make the calculations easier later in our implementations. We then analyze the parsed input data and find the probability for each attribute in all possible label values. This data is then combined with an m-estimator to increase the reliability of our findings.

The process begins by finding the simple average for each positively labeled instance of a specific attribute and repeating that process for the negatively labeled instances. We then multiplied the results in order to find the most likely outcome for each combination of attributes in a data set.

1.3 Evaluation

To evaluate our implementation we first selected 30 random data points from the set to test. After setting those aside and "training" our algorithm we then calculated the P+ and P- values based on the Phat values calculated in the "training" stage. after P+ and P- were calculated. The larger one was then the "predicted" value of the data point. After we figured out the predicted label we checked to see if that matched the actual label of the data point. If it did not we added one to our error count. After all 30 data points were predicted we calculated a confidence interval.

For our first data set, Monks, we found this:

m = 2

Number errors = 10

With 60% confidence errorD(h) = [0.247267036750946, 0.41939962991572]

With 80% confidence errorD(h) = [0.223168473707878, 0.443498192958789]

With 90% confidence errorD(h) = [0.192184606938219, 0.474482059728448]

m = 3

Number errors = 6

With 60% confidence errorD(h) = [0.126970325665978, 0.273029674334022]

With 80% confidence errorD(h) = [0.106522016852452, 0.293477983147548]

With 90% confidence errorD(h) = [0.0802313340922037, 0.319768665907796]

For the second data set, Voter, we found this:

m = 3

Number of errors = 4

With 60% confidence errorD(h) = [0.0712700442499158, 0.195396622416751]

With 80% confidence errorD(h) = [0.0538923233065589, 0.212774343360108]

With 90% confidence errorD(h) = [0.0315495392365286, 0.235117127430138]

m = 4

Number of errors = 1

With 60% confidence errorD(h) = [0.000560263991660828, 0.0661064026750058]

With 80% confidence errorD(h) = [-0.00861619542400748, 0.0752828620906741]

With 90% confidence errorD(h) = [-0.0204145003870096, 0.0870811670536762]

With the final, tic-tac-toe, set we found the following data:

m = 3

Number errors = 9

With 60% confidence errorD(h) = [0.216333997346592, 0.383666002653408]

With 80% confidence errorD(h) = [0.192907516603638, 0.407092483396362]

With 90% confidence errorD(h) = [0.162787755648412, 0.437212244351588]

m = 4

Number errors = 6

With 60% confidence errorD(h) = [0.126970325665978, 0.273029674334022]

With 80% confidence errorD(h) = [0.106522016852452, 0.293477983147548]

With 90% confidence errorD(h) = [0.0802313340922037, 0.319768665907796]

This was just some of the output we had from our program. When running the program we test a few different values for m to see which one will work the best.

1.4 Conclusion

Overall we found that the Naive Bayes Classifier worked fairly well for our data sets. We think that his is because the data is pretty simplistic and it can guess with high probability what the class is based on the attributes. We have learned about conditional probabilities in our other classes but we had never seen an application like this before so it was very interesting to see the results of a useful application like this classifier.

From our confidence intervals above you can see that as we changed the m-estimate our algorithm could become more or less error prone. This is very interesting because like the ANN using this classifier has a virtue of art to it rather than just math. This is because you must try a few different m values in order to find which one is best. We decided to include 3 different intervals for each set because we were not sure how confident we were on the accuracy of our algorithm.

1.5 References

Lecture Slides were used as an example on how to implement this classification method.