Daniel Bis
dmb16f

Assignment 3 Report

1. A brief introduction of the classification method.

   Bayes naïve classifier is based on Bayes' theorem. Bayes theorem is a probability theorem called posterior probability that calculates the probability of hypothesis H given information X – P(H|X). Bayes naïve classifier assumes that the effect of an attribute value of a given class is independent of the values of other attributes (assumption called class-conditional independence).

2. The implementation details including core data structures, algorithms, parameters you chose during the implementation, and discuss why.

   In my implementation, I load the data into a matrix (RxC where R is number of lines of the file and C_0 is a label and C_{1-n} are attribute:value pairs).
   I normalized the data, inserting attribute:value pairs in the missing spots.
   Later I convert the matrix to a matrix in form [[int(label) int(attribute)]..] to make sure that I have numerical data.

   Later I run functions which perform probability calculations. First I use stats_by_class function to compute a mean and standard deviation of each column (attribute) and group these by class. To get the mean and standard deviation I call get_stats where I use python's zip() function to group attributes together and calculate statistics.

   Once I have my parameters calculated, I can pass them into my get_prediction function (in form, *stats: {label: [(mean, st_dev) ...], label2: [(mean, st_dev) ...]}), where I* calculate the Gaussian distribution of each attribute, and combine these following Bayes' Theorem inside of the combained_probability function. Finally, based on the result which is in form of {label1: probability, label2: probability}, do_prediction function returns the label with higher probability.

   Function named summary compares the generated labels with labels given in the file and returns the true positive, false negative, false positive, and true negative counts.

   My code is carefully documented. Comments describe the parameters and return types, as well as the logic behind the functions.

3. All output results you calculated for different datasets;

   Breast Cancer dataset:
   30 98 26 26
   16 62 15 13

```
led dataset:
444 1157 292 194
250 634 149 101
```

4. Personal thoughts and potential improvement.

I compared my output with the output from scikit-learn library and my results surprisingly outperform mentioned implementation. For example, in case of the breast cancer test my true positive count was 16 (test set), when scikit-learn's 12. For led dataset it was 250 returned from my code and 212 from scikit. My false positives and false negatives also returned better results.

In terms of possible improvements to the code, I could try to use different density function instead of the Gaussian (Normal) distribution. Furthermore, probabilities tend to be low or sometimes 0. We could use Laplacin correction to fix the 0s and log likelihood to avoid the underflow potentially caused by multiplication of very small numbers.