# CS 440: naïve Bayes Classification

## Part 1: Digit Classification

### Single Pixels as Features

Our implementation is quite simple. We created an object that would hold matrices for the binary 0 (white) and 1 (black) for each class. These matrices would be the size of an image. This meant that when we encountered a feature, we could simply find that exact location in a specific matrix and increment the count. For example, if we were looking at pixel (1,1) in a training image that represented the digit class 5 and the pixel was black, we would go to the black matrix for class 5 and increment its count. We followed this procedure for every feature of every training image. After doing this, we compared the testing data against the populated matrices for every single class, using the method described in the Assignment handout. For each testing image, we calculated its posterior probability for each class. We labelled it using the highest posterior probability. Our results are shown below.

We chose to use a smoothing factor of 1. This means that we assume that we have seen every value once more than its actual occurrence value. This made our results more accurate than using no smoothing factor. We chose not to use a larger smoothing factor; we did not want to over-count occurrences by more than 1. Using a smoothing factor of 1 means that if a particular feature never occurred for a given class, we could still operate under the assumption that it did occur once. This allows uncommon features to still be represented. Using a larger smoothing factor would have overrepresented features that do not occur very often. We did, however, build functionality to play around with smoothing factor. We made it possible to vary smoothing factor very easily. In our implementation, smoothing factor can be specified, which allows us to test different smoothing factors very easily. We found the best results with a smoothing factor of 1, as per our hypothesis. We want to represent all features, even if they occur uncommonly, but we do not want to over-represent uncommon features, which larger smoothing factors would do.

#### Results, as referenced above

Our overall accuracy rate for the digit data set is 77.1%

Below are the classification rates for each digit. The format is as follows:

*(digit class, classification rate – truncated after 3 decimal points)*

(0, 0.844)

(1, 0.962)

(2, 0.776)

(3, 0.790)

(4, 0.766)

(5, 0.673)

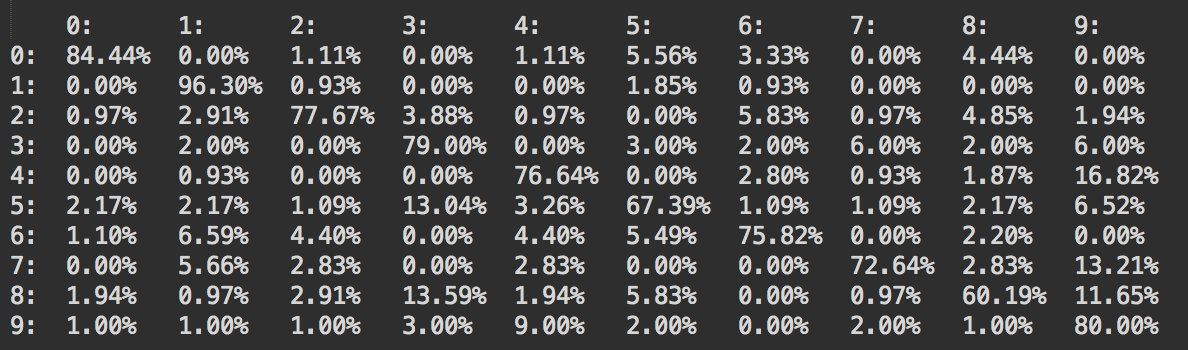
(6, 0.758)

(7, 0.726)

(8, 0.601)

(9, 0.800)

Below is the confusion matrix. Percentages are truncated after 2 decimal points.



We also looked at odds ratios for the following sets of numbers: [(7,9),(4,9),(5,3),(8,3)]. The results are below.

### Extra Credit for Part 1

We experimented with using ternary features. This is for bonus points. Our implementation was similar, but we had matrices to represent white, gray, and black rather than just white and black. Our results reflected a small improvement.

Our overall accuracy improved to 77.6%.

Below are the classification rates for each digit. The format is as follows:

*(digit class, classification rate – truncated after 3 decimal points)*

(0, 0.833)

(1, 0.953)

(2, 0.766)

(3, 0.800)

(4, 0.775)

(5, 0.684)

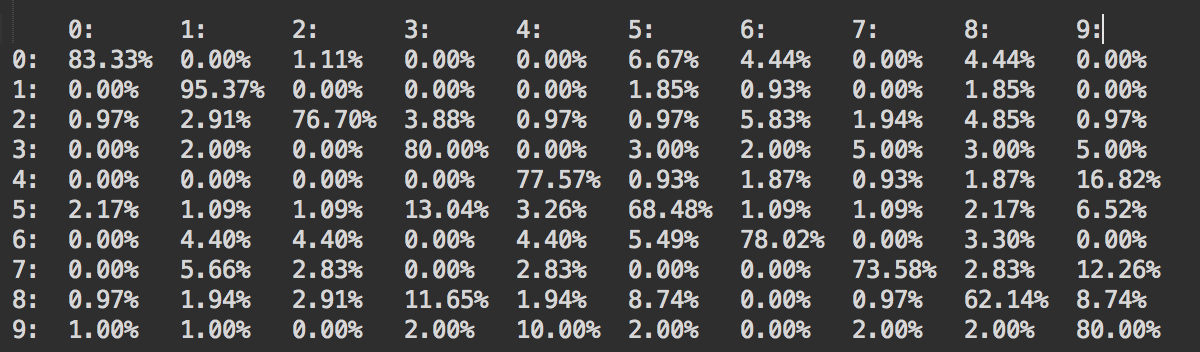
(6, 0.780)

(7, 0.735)

(8, 0.621)

(9, 0.800)

The confusion matrix is shown below.



It appears that ternary features did not make a significant improvement upon classification. However, we do believe that weighting gray values differently could be worth further exploration.

We applied our Naïve Bayes Classifier to the face data set. The implementation was very similar to the digit data set, except that there were only two classes and that there were more features per image. This is for bonus points.

Our overall accuracy was 90.6%.

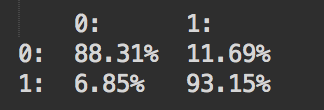
Below is the classification rate for each class. The format is as follows, with 0 being not face and 1 being a face:

*(face class, classification rate – truncated after 3 decimal points)*

(0, 0.883)

(1, 0.931)

The confusion matrix is shown below.



This whole section was for bonus points.

## Part 2: Document Classification

### 1.1