**Project: Face and Digit Classification**

Kevin Nehrbauer RUID: 169005809

Christopher Woolley RUID: 170007112

Naive Bayes

Using the code skeleton provided by Berkeley, Naïve Bayes is implemented under the trainAndTune and calculateLogJointProbabilities methods. In trainAndTune, our code creates three counters to calculate the prior probability, conditional probability, and counts (which is the total data set used to calculate conditional probability). Using a for loop through the training labels gets us the prior probability, and a nested for loop going through each feature contained in training data, adding one to counts([feature, label]) each time, and adding one to conditional probability([feature, label]) if the label is equal to 1 for that particular feature until everything in the dataset has been iterated through. Once the prior is normalized (number of training instances with label y over the number of training instances) it then uses another nested for loop that goes through each value [(feature, label)] in counts and conditional probability, adds 1 to each conditional value and 2 to each counts value (since the only possible feature values can be 0 or 1. To get the final conditional probability values, a for loop sets each conditional probability[i] is set to (conditionalProb[i] / counts[i]). Once this is done, the Berkeley code calls calculateLogJointProbabilitites, which takes for each item i the log(prior probability) + the sum of the logs of each feature’s conditional probability. Overall, Naive Bayes resulted in somewhat accurate results using little of the training data as shown in the graphs below. As the amount of training data increases, the accuracy for both digit and face classifying increases, displaying a learning curve.



