Project completed by Andrew Gallagher (arg213), Stephen Kuo (stephkuo), and Lu Liu (ll685)

Image Processing and Feature Extraction

For this project, we used the skeleton code provided on Berkley's website.

Naive Bayes Classifier

1). Features
For our Naïve Bayes implementation, we used pixel features. For each pixel, it can be either a value of 0(denotes that a pixel is white), or a value of 1(denotes that a pixel is black). Returns the probability that it is a certain digit.

$$P(F_1, \ldots, F_n, Y) = P(Y) \prod_i P(F_i|Y)$$

2). Smoothing
We used Laplace Smoothing, which is adding a constant *k* to every observation value:

$$P(F_i = f_i|Y = y) = \frac{c(f_i, y) + k}{\sum_{f_i' \in \{0,1\}} (c(f_i', y) + k)}$$

3). Train and Tune
We need:
P(Y = y) = (number of data with label Y = y in training set) / (total number of training data)

4) Classify
This was included in the Berkley code; we compute the log probability of P(*y*|f1,…,fn). The label with maximum probability will be our guess.

5) Calculate Log Joint Probabilities

We needed to also computer the log probabilities which have the same argmax because just multiplying many probabilities together would result in an underflow

Experiment Time

Digits Classification:
We began by using 10% of the total training data, and then continuously upping the percentage. Next was 20%, then 30%, 40%, 50%, 60%, 70%, 80%, 90%, and lastly 100%.
Below is a table detailing our findings:

Percentage of Training Data Used	Accuracy
10%	65%
20%	69%
30%	69%
40%	73%
50%	76%

60%	78%
70%	79%
80%	83%
90%	81%
100%	78%

Percentage of Training Data Used	Time Taken To Train(seconds)
10%	13.79
50%	14.58
100	15.25

Faces Classification:
We began by using 10% of the total training data, and then continuously upping the percentage. Next was 20%, then 30%, 40%, 50%, 60%, 70%, 80%, 90%, and lastly 100%.
Below is a table detailing our findings:

Percentage of Training Data Used	Accuracy
10%	74%
20%	77%
30%	85%
40%	85%
50%	84%
60%	86%
70%	86%
80%	86%
90%	88%
100%	89%

Percentage of Training Data Used	Time Taken To Train(seconds)
10%	19.66
50%	23.89
100%	29.74

Perceptron Classifier

1). Features
For our Naïve Bayes implementation, we used pixel features. For each pixel, it can be either a value of

0(denotes that a pixel is white), or a value of 1(denotes that a pixel is black). Returns the probability that it is a certain digit.

2). Weights
Weights start at 0 and get modified as we go along.

3). Training
For training our program, a max iteration # is set. Once that # is reached, the training stops. For each iteration, we loop through the training data, and for each datum, a list of numbers are computed corresponding to the labels using this equation:

$$f(Xi,w) = W0 + W1\Phi 1 + \ldots + Wk\Phi k$$

Then the label with the highest f(*Xi,w*) is our guess. If guess is the label, it means our program predicted it correctly and no changes are necessary. Otherwise if guess != label, the weight of the correct label is too small and weight list for our guess is too high. Thus modifications must be made:

For x = 1, 2,…k : weights[label][x] += Φ(datum)

W0 += 1

For x = 1, 2,…k : weights[label][x] -= Φ(datum)

W0 -= 1

This process is repeated until nothing is changed within an iteration, or until the max iteration is reached.

Experiment Time

Digits Classification:
We began by using 10% of the total training data, and then continuously upping the percentage. Next was 20%, then 30%, 40%, 50%, 60%, 70%, 80%, 90%, and lastly 100%.
Below is a table detailing our findings:

Percentage of Training Data Used	Accuracy
10%	48%
20%	62%
30%	69%
40%	69%
50%	75%
60%	82%
70%	76%
80%	66%
90%	76%
100%	75%

Percentage of Training Data Used	Time Taken To Train(seconds)
10%	4.47
50%	15.1
100%	28.39

Faces Classification:
We began by using 10% of the total training data, and then continuously upping the percentage. Next was 20%, then 30%, 40%, 50%, 60%, 70%, 80%, 90%, and lastly 100%.
Below is a table detailing our findings:

Percentage of Training Data Used	Accuracy
10%	62%
20%	76%
30%	72%
40%	81%
50%	79%
60%	87%
70%	80%
80%	80%
90%	82%
100%	85%

Percentage of Training Data Used	Time Taken To Train(seconds)
10%	4.15
50%	10.46
100%	17.96

Kowalski, ANALYSIS!

In all our experiments, our program can perform with an accuracy of over 70% given enough training data.

The time the program spends on the training process is proportional to the amount of training data given. Thus, the more training data that we provide the program, the longer it takes to finish the training process.

For accuracy, it is proportional to the amount of training data provided AT FIRST. However, after a certain amount of training data is provided, the accuracy converges to a certain point

and will not increase much if additional training data is provided. In fact, the accuracy may drop when more training data is given due to overfitting.

Conclusions

Essentially, more training data does not guarantee higher accuracy. Accuracy will converge around a certain point depending on what algorithm/feature is being used. Thus, when training our program, it is good to do some experiments to find the optimal amount of training data needed to reach that point of convergence. Otherwise we would waste time on the training process without gaining more accuracy.