Project 2

Face and Digit Classification

Zain Ali (zaa23)
Michael Saunders (mbs189)
Hamza Inayat (hi78)

Rutgers University
Summer 2020

1 Implementation

1.1 Setup

For this project, we utilized skeleton code from Berkeley's CS188 course developed by Dan Klein and John DeNero. We completed the project using Python 2.7 due to the version the skeleton code was. The algorithms completed for this project were Perceptron and Naive Bayes. Information about the algorithms is from the professor's lectures on Sakai and two YouTube videos from another Rutgers Professor, Abdeslam Boularias, who teaches this class called *Part I: Perceptron* and *Part II: Naive Bayes Classifier* .

1.2 Code

Data was parsed and features were based on 1x1 pixel values. Both face and digit data consist of the characters ' ', '+', and '#', these were converted to 0, 1, and 2, respectively for use in the algorithms. Naive Bayes used a smoothing value of 2 and the weights for Perception were initialized as 0.

1.3 Data

The given data sets consisted of the following number of items:

	Digits (0-9)	Faces (0-1)
Training	5000	451
Validation	1000	301
Testing	1000	150

2 Results

Each Algorithm was run with both the Faces data and the Digits data. For purposes of data collection, each dataset was run on each algorithm with increasing amounts of training data used (10%, 20%, 30%, ... , 100%). In order test the algorithm under random data conditions, for each percentage point, the algorithm was run 5 times using randomly selected data to fill that percentage. For each percentage point, we collect the average accuracy, average time, and standard deviation of the 5 trials.

2.1 Accuracy

For both algorithms we calculate what the accuracy is for the model given increasing amounts of training data. For each of the algorithms we see that the more data used to train, the higher the accuracy becomes (See Figure 1&2). There is also a trend of the the faces dataset yielding a higher average accuracy when compared to the digits dataset. For the Perceptron using 100% of the training data, the Faces model had a 88% accuracy and the Digits model with a 82.02% accuracy. When using Naive Bayes at 100% training data, the Faces has a 90% accuracy and the Digits had an accuracy of 76.6%.

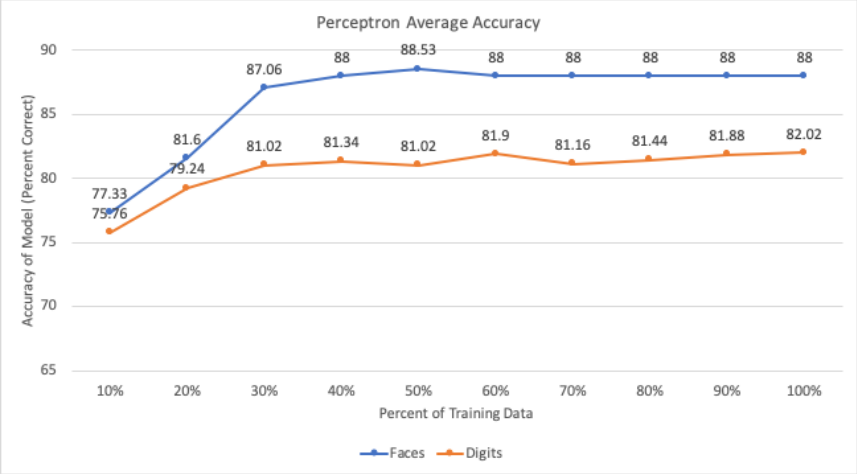

Figure 1: Perceptron average accuracy of 5 trials with faces and digits datasets

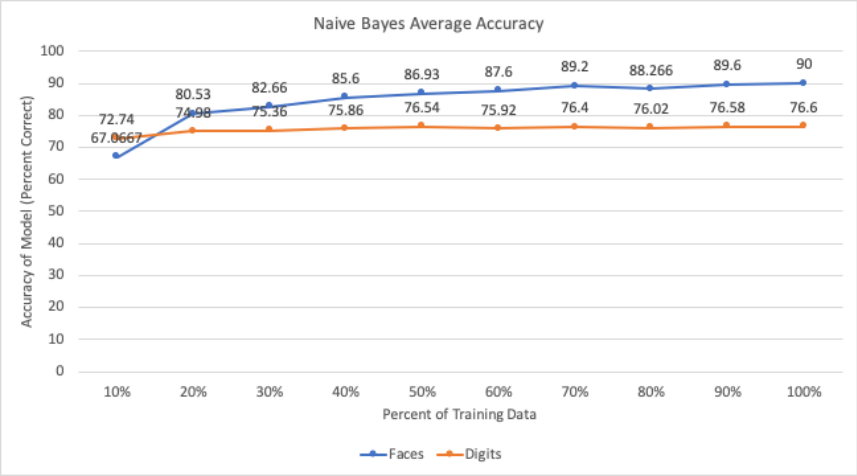

Figure 2: Naive Bayes average accuracy of 5 trials with faces and digits datasets

2.2 Standard Deviation

Among the data collected, there was also the standard deviation of the 5 trials in terms of accuracy. For all of the trial blocks, there was a general trend of the standard deviation getting smaller as the training data used increased. For Naive Bayes, both datasets had it where at 100% data used, the standard deviation was 0. This was not the case with the Perceptron where the Faces model capped out at 0 SD at 60% data used and the Digits data never reaching 0, even at 100% (See Figure 3&4).

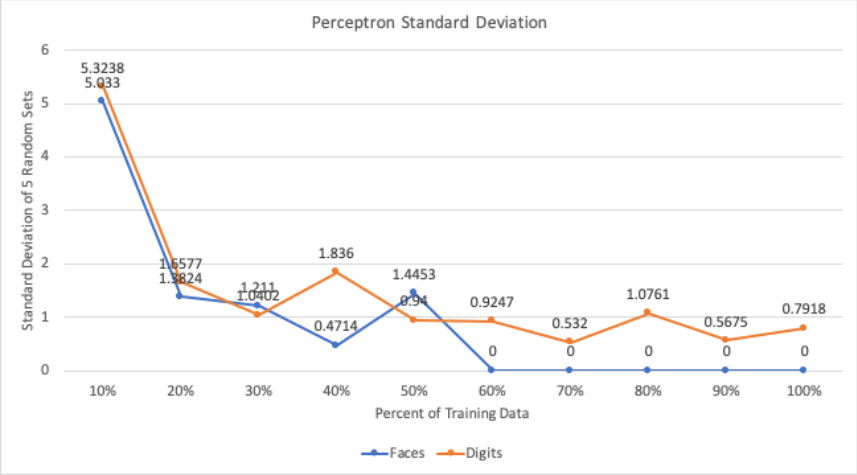

Figure 3: Perceptron standard deviation of accuracy for 5 trials with faces and digits datasets

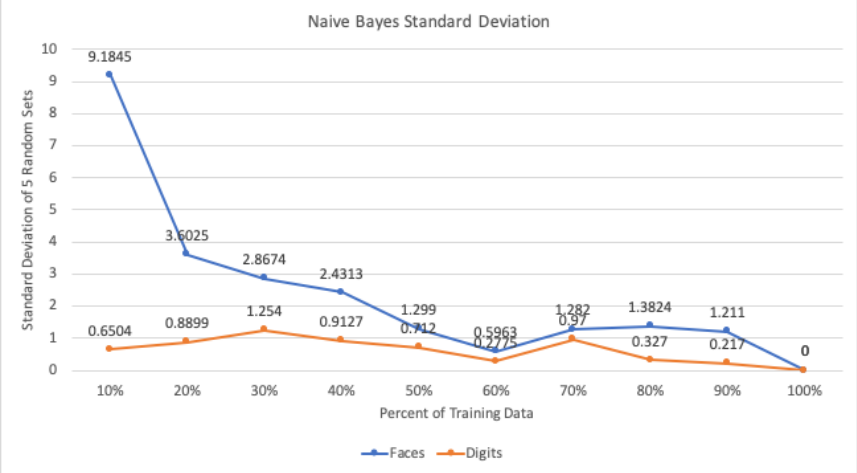

Figure 4: Naive Bayes standard deviation of accuracy for 5 trials with faces and digits datasets

2.3 Time

Another measurement that was used to understand the algorithms was run time. All trials were run locally using Python 2.7. However, these times are relative to the machine running then. Since all trials were run on the same machine, we can assume they are consistent.

When looking at the Perceptron, there was a general trend upward as the data used increased. We see that the digits dataset takes much longer on average than the faces data set. As a reminder the faces training dataset consists of 451 items and the the digits having 5000. The Naive Bayes was opposite with the faces dataset taking longer. What is also seen is that both sets were less divergent (See Figure 5&6).

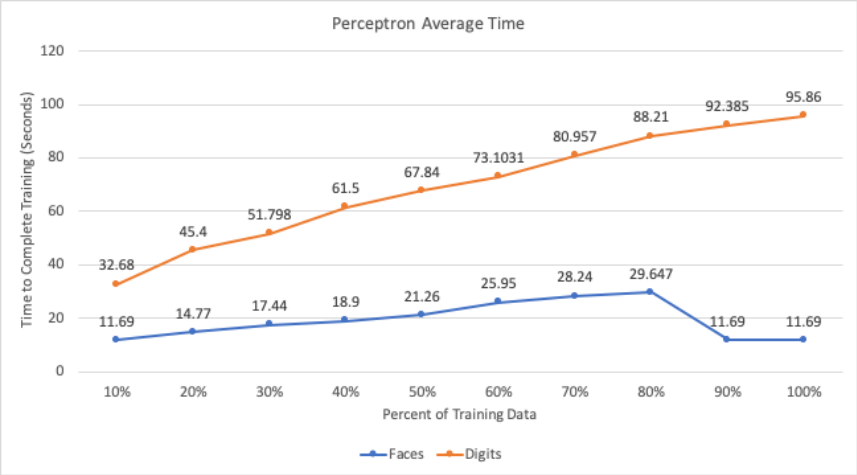

Figure 5: Perceptron average time of 5 trials with faces and digits datasets

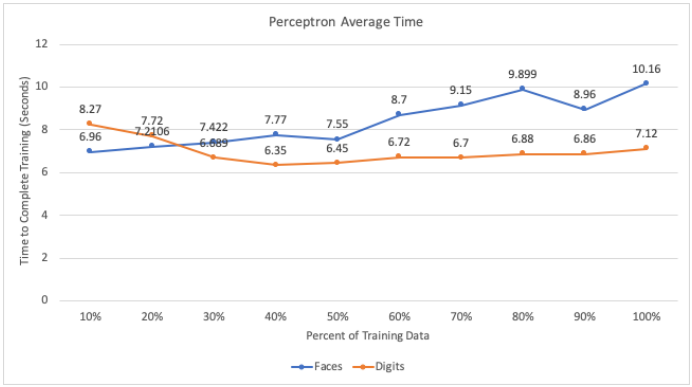

Figure 6: Naive Bayes average time of 5 trials with faces and digits datasets

3 Discussion

As we look towards the Average accuracy, we should also look at standard deviation. As the percentage of data used increased, we can see that the accuracy of the model increased while the standard deviation of the accuracy decreased. This can mean that the more data we used, the more "confident" out accuracy is. There is less variant with minor changes in data. We could also say that there was the same training data set used for each percentage point, as we used higher percentages of the data, there was less variation in the data given. This is why at 100% data used, we see in most cases that the standard deviation end up being 0. This can be because at 100% there is randomness to the data used, and that the data set will be the same for each trial (See Figures. 1-4).

When we look to the time, we can see that there is an increase in time as the percentage points increase. For Perceptron, the time for digits peaked at almost 96 seconds. This is compared to the faces data peaking at 29.6 seconds. When looking at this, we can come to the conclusion it is because of dataset size. Faces had almost 1/10th the amount of data that the digits has. In addition to that, there were 10 classes for digits while only two (1 or 0) for faces. This can contribute to increased run time for Perceptron digits (See Figure. 5).

As for Naive Bayes, the runtime fairly constant throughout the percentage points. This can be attributed to how the Naive Bayes algorithm works compared to the Perceptron. While the Perceptron needs to have multiple iterations, or epochs, Naive Bayes run for only one iteration. Because of this, some of the variation can be attributed to the machine simply performing slowing for that trial in addition to increasing data percentages used. Naive Bayes takes significantly less time compared to the Perception (See Figure. 5&6).

4 Conclusion

In the end, through this project, we were able to learn about two learning algorithms. Things that we observed were the increase of accuracy as a function of the percent of training data used. This is also in line with how the standard deviation goes down when testing with more data. Another observation made was how the time for perception increased with the more data used. This is in contrast to what was seen with Naive Bayes where the time taken was less than Perceptron. Some of the time differences for Naive Bayes could be attributed to the machines the algorithm was running on.

Overall, the data we got was expected and the results were good. When it comes to more epochs for Perceptron, it was hard to decide whether to add more or just to run with 3. For larger data percentages more epochs would just end up being cut off by early convergence, so it was decided to just use 3 to keep our independent variables less. When it came to Naive Bayes, when experimenting with randomized initialization of the weight vector lead to inconsistent results. At times it would be better, and other times worse. This is initialization of 0 was chosen for data collection. As for the smoothing value, 2 was chosen for the purpose of data collection. However, a smoothing value of 1 could also be effective since the range of features were from (0-2). Anything more than 2 seemed to give worse results. Perhaps the biggest improvement that could be made would be have feature extraction based on a larger grid rather than just individual pixel features. This was not implemented so we cannot definitely say that it would improve accuracy.