Project 2 Report

Group 29: Feiyu Zheng(fz114) & Xueyang Chen(xc186)

August 10, 2020

1. How do we do feature extraction?

We use each single pixel in the graph as a feature, and the name/key for each feature is its location. If it is a white space, we assign the value of this feature to 0, otherwise 1.

2. How do we implement each algorithm?

Naive Bayes:

Since we need to use Bayes Rule equation to predict the probabilities of each outcome(label), we have to implement it in the coding. For Baye Rule,
(a) probability of outcome(specific label) given a feature set of a sample is equal to the product of (b) probability of outcome(specific label) and (c) probability of this data given outcome is a specific label divided by (d) the probability of the data.

Because we only need to know which outcome(label) has the greatest probability, we compare the value of each outcome(label) for the same data(sample). Since (d) the probabilities of the data are always the same, we just need to compare the value of (b) probability of outcome(specific label) times (c) probability of this data given outcome is a specific label.

Because multiplying many probabilities together often results in underflow, we will instead compute log probabilities.

For (b) probability of outcome(specific label), we use the number of samples with specific label divided by total number of samples. Then, calculate its log value.

For (c) probability of this data given outcome is a specific label, it is the same as the product of (e) the probability of each feature given a specific outcome(label). We calculate each (e) probability of a single feature given a specific probability(label), and then multiply their log values.

For (e) probability of a single feature given a specific probability(label), we count the total times a specific feature has value 0 with a specific label, and divided by the total number of samples with this specific label. Since we only have two possible values for each feature(0 and 1), if a specific feature has value 1, then we use 1 minus the probability we have above to get the answer. Lastly, we convert the value to log value.

We sum all log values we get above to get our log probability of a specific label for a data. The label with greatest probability is our final prediction.

When we do the training process, we use a smoothing parameter k to improve the accuracy of prediction. We have a set of 10 different k values, and each iteration runs with one k value. After 10 iterations, we assign k to the value with best accuracy.

Perceptron:

For all training data, we pick them one by one to do the following training process:

(1) For each training data, we have A features vectors and B labels. Then we initialize the weight as a BxA matrix which contains small values (0.5 in our project). Multiplying the weight matrix by the features vectors will yield a vector that contains a value corresponding to each label. Among these values, we choose the label with the largest value as our prediction

(2) We compare our prediction with the actual answer and take different actions depending on the comparison result.

  (a) If the label we predict is right, then we don't modify the weight and jump to the next picture.

  (b) If the label we predict is not the same with the actual label,

    (i) we increase the weight value used for the real label if the current result for the real label is less than zero.

    (ii) we decrease the weight value used for the prediction label if the current result for that label is greater than zero.

(3) We keep doing the above two steps until the algorithm reaches the iteration limits we predefined or there is no weight value change for all training data.

3. The learning curve for each case.

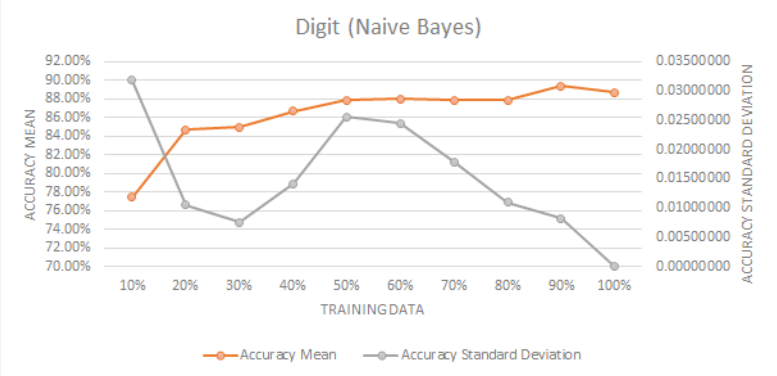

Figure 1: Accuracy of Using Naive Bayes on Digit Data

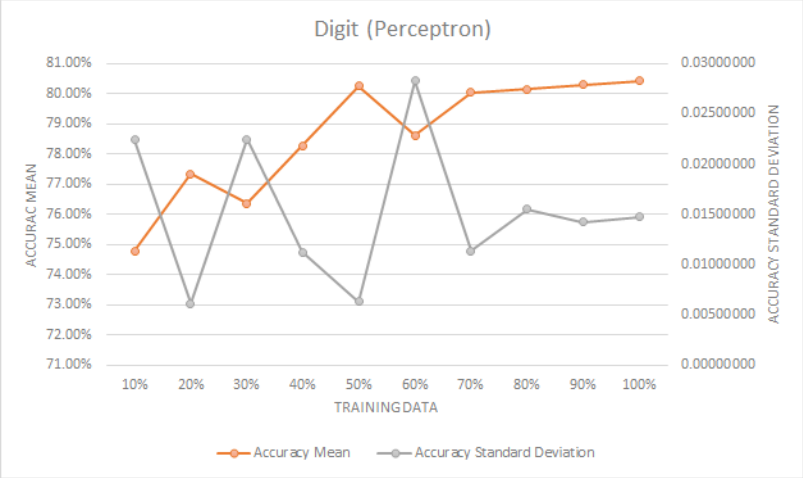

Figure 2: Accuracy of Using Perceptron on Digit Data

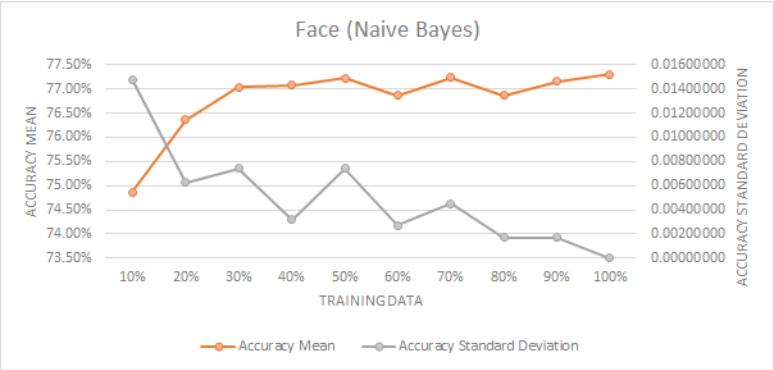

Figure 3: Accuracy of Using Naive Bayes on Face Data

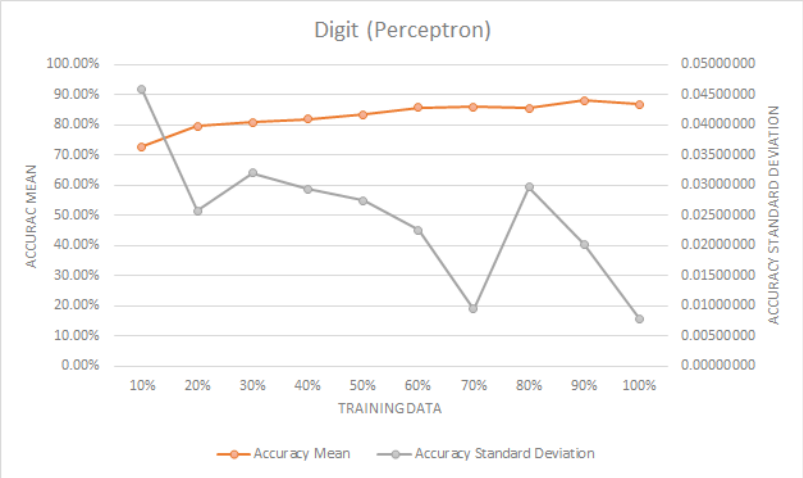

Figure 4: Accuracy of Using Perceptron on Face Data

As above Figure 1-4 show, we can see that the accuracy goes up as we are using more and more training data and we can also see a trend that the amount of data needed to increase the

accuracy is increasing as the accuracy keeps increasing. As we are getting higher and higher accuracy, it's harder and harder to increase the accuracy by adding more training data.

The following Figure 5-7 are the visualization of trained weight in the perceptron algorithm. With 100% training data, the algorithm can already classify some labels by some points and know the general shape of a type of images.

Figure 5: Perceptron Weight Visualization of Digit 1

Figure 6: Perceptron Weight Visualization of Digit 3

Figure 7: Perceptron Weight Visualization of Face

4. Our observation on statistics.

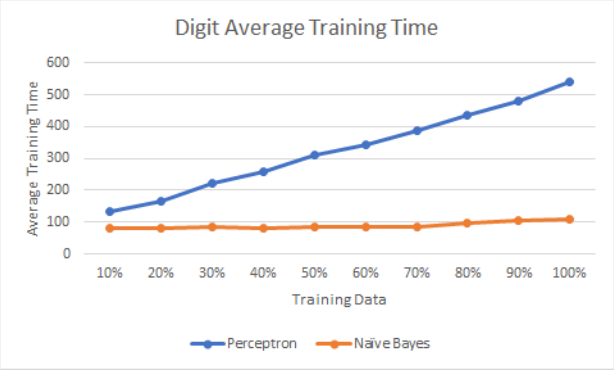

Figure 8: Training Time of Perceptron and Naive Bayes When Using Digit Data

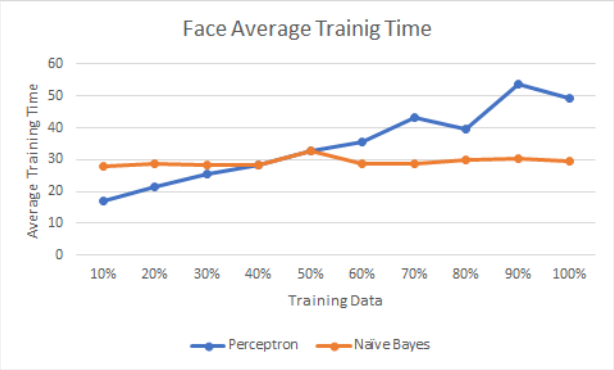

Figure 9: Training Time of Perceptron and Naive Bayes When Using Face Data

Figure 8-9 show the time needed for training. We can see that as the number of training data gets larger and larger, the time needed for perceptron algorithms increases much faster than the naive bayes.

In Figure 1-4, the gray lines show the standard deviation as the number of data points increases during training. We can see a general pattern that as the size of training data increases, the prediction error is getting smaller and smaller and for Naive Bayes, the prediction error is always 0 when using 100% training data.

5. What can be improved in the future if it does not work well or does not have a high accuracy

We can choose the better features which are more related to the result. For instance, the number of non-white space in a row or column, or the location of the farest gray space in each line.