AI Final Report: Naive Bayes and Perceptron Classifiers

Dayyan Hamid NETID: dh820

December 2024

1 Introduction

This project focuses on building and testing two classification algorithms, Naive Bayes and Perceptron. These were for digit recognition (10 classes) and face classification (2 classes). The goals were to design features for the images, implement the classifiers, and compare their performance based on accuracy and training time. Naive Bayes was especially interesting for me because I recently had to use it in my machine learning class to correctly identify spam emails. I worked on calculating the likelihood of an email being spam based on its contents. For example, the algorithm determines the probability of an email being spam given certain words or phrases in the email. Applying it here to image data was honestly really cool, I enjoyed it! The Perceptron algorithm was new for me. It required iterative updates to weights and biases, which made it feel more dynamic compared to Bayes. Through this project, I got to directly compare these two approaches and understand their strengths and weaknesses for different types of tasks.

2 Feature Extraction

The feature extraction involved loading images and converting them into binary matrices where pixels were either 0 (background) or 1 (part of the digit/face). Each image was processed line by line to ensure correct representation.

I found a problem in the data file and that the line separator was being counted as a character, which messed up the creation of the binary matrix. Fixing this required carefully ensuring that each line in the file was correctly read and stripped of unnecessary characters.

The final binary feature extraction code for images is here:

```
for i in range(num_images):
    #we need to eaxtract the image lines here
    image_lines = lines[i * image_height: (i + 1) * image_height]
    #this is a list of lines for each image, as in image one is from 0
```

```
        #(0 * image_height = 0) to (1 * image_height)
        binary_image = []
        for line in image_lines:
            binary_line = []
            for char in line:
                if char in ['#','+']:
                #nothing is 0 and anything is 1 in the image
                    binary_line.append(1)
                else:
                    binary_line.append(0)
            binary_line = binary_line[:image_width] +
            [0] * max(0, image_width - len(binary_line))  # Ensure the correct width
            binary_image.append(binary_line) #line by line added
        images.append(binary_image) #this numpy array contains the binary for images
    images = np.array(images) #this is the numpy array of images
```

3 Classifiers Implementation

3.1 Naive Bayes Classifier

The Naive Bayes classifier was implemented to calculate class priors and feature probabilities. During the initial implementation, the results were poor, with warnings for division by zero or logarithm of zero. After researching, I added Laplace smoothing (using an α value of 1) to resolve this issue:

$$P(X|C) = \frac{\text{sum of feature values} + \alpha}{\text{number of samples in class} + 2\alpha}$$

This significantly improved the results.

3.2 Perceptron Classifier

The Perceptron classifier used a weight matrix and a bias vector for training. The weights were updated iteratively during training whenever the classifier made an incorrect prediction. The key update rule was:

$$w_c = w_c + \text{adjustment} \times \text{sample}$$

Where the adjustment was +1 for the correct class and -1 for incorrect classes. This approach allowed the Perceptron to improve over time.

4 Evaluation Methodology

Both classifiers were evaluated using a subset of training data, ranging from 10% to 100%, in increments of 10%. For each percentage, the classifiers were trained and tested 5 times to compute the mean accuracy and standard deviation.

5 Results and Discussion

The performance of the two classifiers was compared based on training time and accuracy.

5.1 Training Time

The training time as a function of the training size is shown in Figure 1. Naive Bayes was significantly faster than Perceptron because it calculates probabilities directly without iterative updates. Perceptron, on the other hand, requires multiple epochs of weight updates, making it slower as the training size increases.

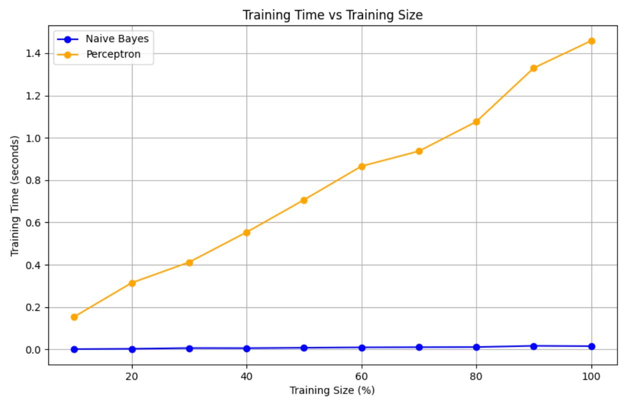

Figure 1: Training Time vs Training Size

5.2 Accuracy

The accuracy as a function of the training size is shown in Figure 2. Both classifiers improved with larger training sizes, but the Perceptron consistently outperformed Naive Bayes in terms of accuracy. For the digit recognition task, the Perceptron achieved a maximum accuracy of 81%, while Naive Bayes peaked at 77%. For face classification, Perceptron and Naive Bayes both performed well, achieving similar accuracies of around 88%.

5.3 Insights

The following insights were observed from the comparison between the Naive Bayes and Perceptron classifiers:

- **Training Time:** Naive Bayes is much faster because it doesn't require multiple passes (or iterations) through the training data.

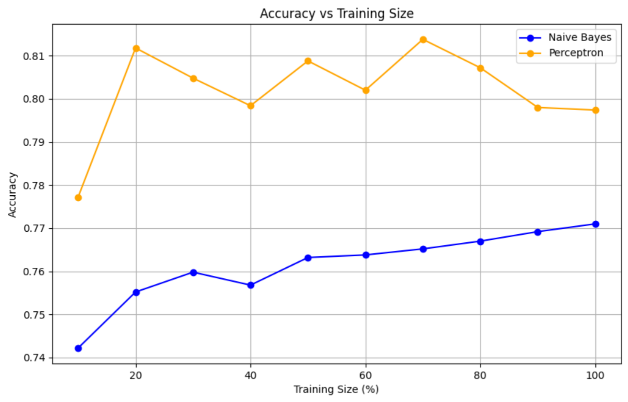

Figure 2: Accuracy vs Training Size

- **Accuracy:** Perceptron is more effective for the digit dataset, due to its ability to iteratively adjust weights for more complex patterns.
- **Face Classification:** Both classifiers performed comparably, showing that the simpler Naive Bayes model is good for binary classification tasks like faces.

6 Challenges

- **Data Loading Issue:** Initially, the binary matrix creation was incorrect because of line separators being misinterpreted. I fixed this with clean line reads and carefully handling newline characters.
- **Naive Bayes Division by Zero:** Early in the project I kept getting division by zero errors, and getting bad results .07 accuracy. Adding Laplace smoothing fixed this issue
- **Iterative Testing:** Testing multiple percentages of the training data and tracking standard deviations was difficult, my evaluation fucntion had to work for both and was tedious trying to fix it when there were errors.

7 Conclusion

This project successfully implemented and compared Naive Bayes and Perceptron classifiers for digit and face classification tasks. Perceptron achieved higher accuracy, but Bayes was significantly faster. These show which algrothim is used best where. The Perceptron is better for digit recognition because it gets a higher accuracy and handles the multi class (10 classifications) better than

Bayes. For face classification, both Naive Bayes and Perceptron perform almost the same. But Naive Bayes is faster and simpler, making it a better choice for binary tasks.