CS 440 Project 2: Face and Digit Classification

Geet Purohit 188001760 and Miguel Ranjo 188002839

December 14, 2021

Introduction

For this project, we built three implementations Perceptron, Naive Bayes, and MIRA. Below we write about our implementation, our performance, comparison of performances, and analysis.

Implementation Description

Peceptron

Perceptron generates a line that separates the labeled data to classify. As so, perceptron is called a linear classifier. It is mathematically expressed as:

$$f(x) = \begin{cases} 1 & w \cdot x + b > 0 \\ 0 \end{cases} \tag{1}$$

Here, w is our weight vector, x is our input vector, and b stands for bias. This bias is what pushes the decision boundary from the origin.

Naive Bayes

Naive Bayes works off of Bayes Theorem which is:

$$P(C_i|D) = \frac{P(C_i)P(D|C_i)}{P(D)} \tag{2}$$

Naive Bayes calculates the conditional probability for each feature along with laplace smoothing with some k. Guesses are made using:

$$argmax_{Ci} logP(C_i) + logP(D|C_i) \tag{3}$$

The log-joint probability is given by the formula:

$$logP(C_i|D) = logP(C_i) + logP(D|C_i) \tag{4}$$

MIRA

MIRA is an extension of Perceptron and for this algorithm, feature weights are learnt with additional loss function and margin constraint. At classification level, MIRA lines up with perceptron, but the way it trains differs. MIRA makes use of validation data and chooses from different weights based on which one gives the best accuracy.
Implementation:
1. With each C value (0.002, 0.004, 0.008), we iterate through these following steps.
2. Take each datum and iterate over all labels to find the best match/guess.
3. If the match/guess is wrong, we then have to modify our weight matrix. The update rule makes use of the C value. This step varies with the perceptron.
4. We choose the weight matrix for C, and of which we get max accuracy.

Features Implemented

1. Number of black pixels (0 in the matrix) as we use them to refer to inactive pixels
2. Splitting the image into four regions by x and y central axis. From there, we get the percentage of active pixels for each region.
3. Number of breaks in horizontal and vertical spaces

Individual Performance

1. For digit classification, we have
2. For face classification, we have
Each of the algorithms have been trained on a range of data points and we observe the following.

Perceptron

Digits			
Percentage of Total Data	Training Data	Accuracy Percentage	Time Taken To Train (Seconds)
10	500	56.6	4.61
20	1000	72.3	9.06
30	1500	71.5	13.90
40	2000	75.6	18.47
50	2500	76.5	22.82
60	3000	81.2	27.62
70	3500	77.6	33.20
80	4000	79.4	38.31
90	4500	77.9	41.93
100	5000	81.5	46.07

Faces			
Percentage of Total Data	Training Data	Accuracy Percentage	Time Taken To Train (Seconds)
10	45	58.7	0.65
20	90	56	1.12
30	135	70	1.73
40	180	78.7	2.22
50	225	71.3	2.83
60	270	84	3.46
70	315	80	3.93
80	360	87.3	4.42
90	405	88	5.14
100	450	86.7	5.64

MIRA

Digits			
Percentage of Total Data	Training Data	Accuracy Percentage	Time Taken To Train (Seconds)
10	500	55.7	6.27
20	1000	61.3	11.41
30	1500	74.4	17.28
40	2000	78.5	22.87
50	2500	73.8	28.27
60	3000	78.8	33.89
70	3500	79.3	40.16
80	4000	79.3	43.97
90	4500	75.5	49.56
100	5000	80	57.49

Faces			
Percentage of Total Data	Training Data	Accuracy Percentage	Time Taken To Train (Seconds)
10	45	54.7	0.90
20	90	74	1.65
30	135	62.7	2.78
40	180	62.7	3.19
50	225	76.7	3.94
60	270	79.3	4.92
70	315	79.3	5.51
80	360	84	6.72
90	405	83.3	7.30
100	450	80.7	7.77

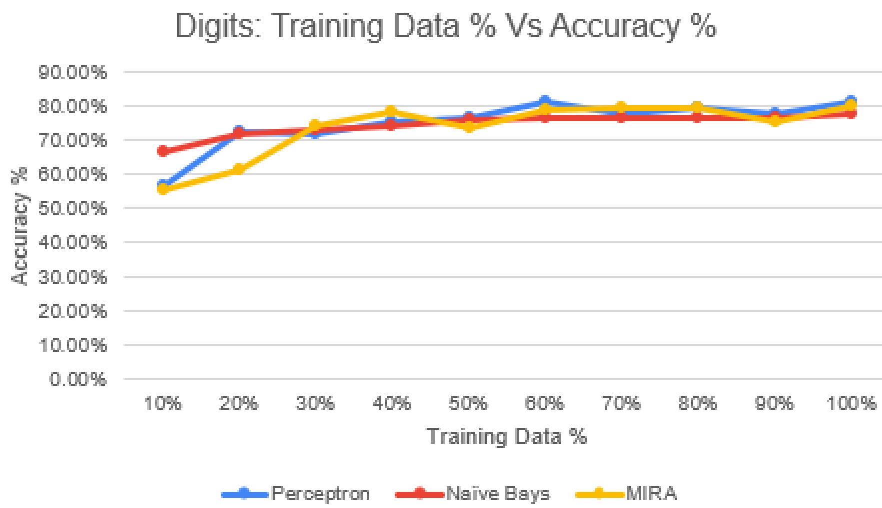
Naive Bayes

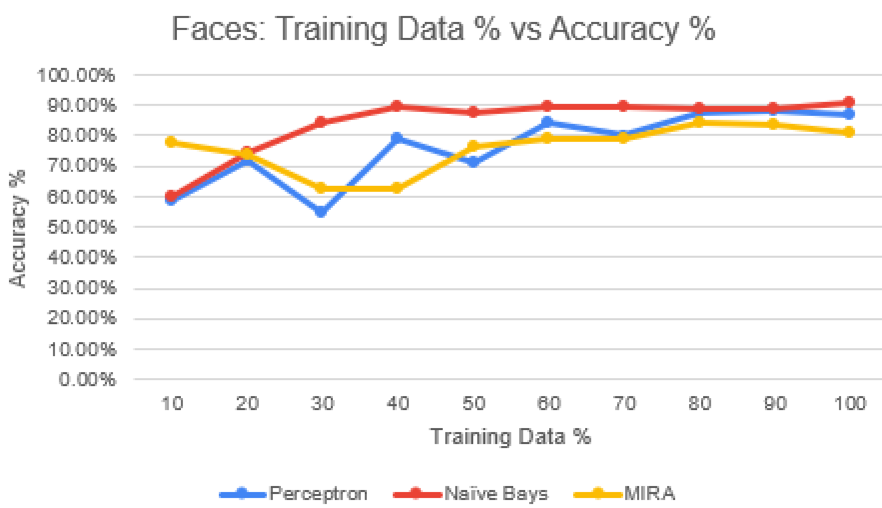
Digits			
Percentage of Total Data	Training Data	Accuracy Percentage	Time Taken To Train (Seconds)
10	500	66.7	14.31
20	1000	71.9	15.45
30	1500	72.8	15.66
40	2000	74.4	16.51
50	2500	76	18.26
60	3000	76.3	16.37
70	3500	76.3	17.51
80	4000	76.7	17.53
90	4500	76.7	18.43
100	5000	77.6	18.99

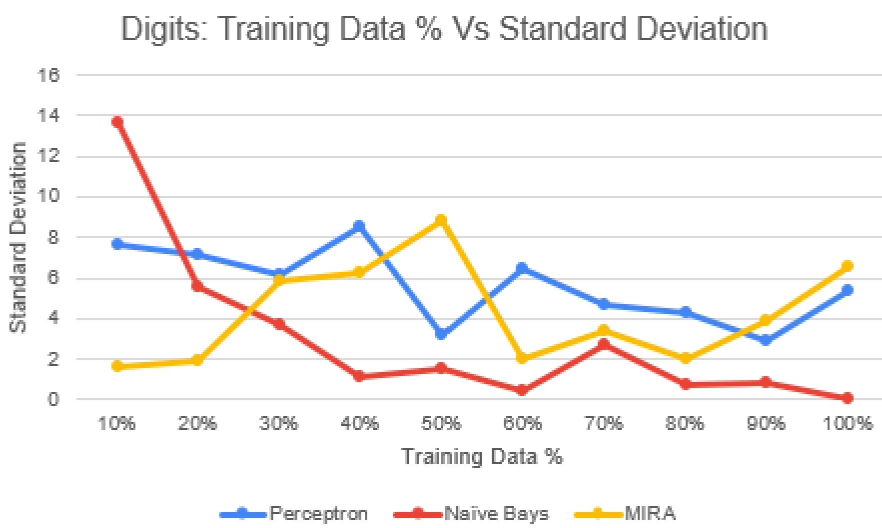
Faces			
Percentage of Total Data	Training Data	Accuracy Percentage	Time Taken To Train (Seconds)
10	45	60	2.80
20	90	74.7	3.24
30	135	84	4.21
40	180	89.3	4.43
50	225	87.3	5.3
60	270	89.3	6.02
70	315	89.3	6.01
80	360	88.7	7.19
90	405	88.7	8.45
100	450	90.7	9.16

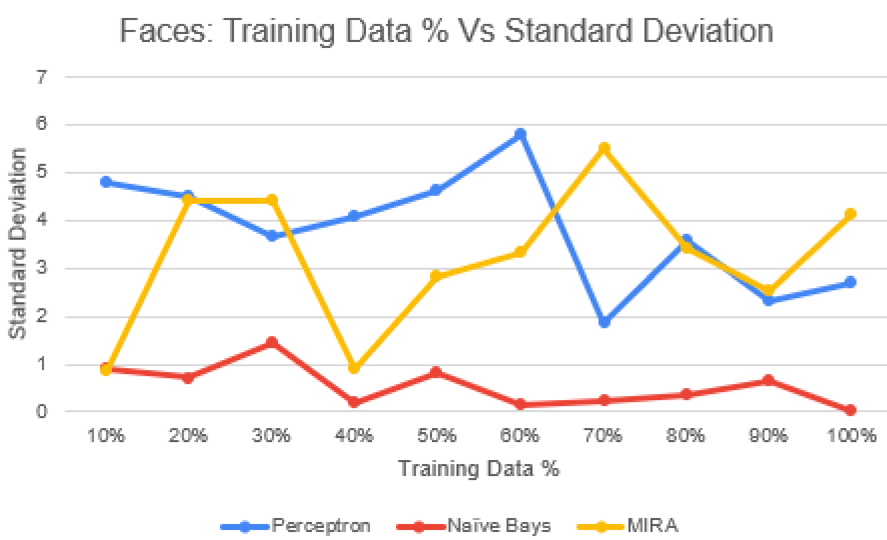
Compared Performance

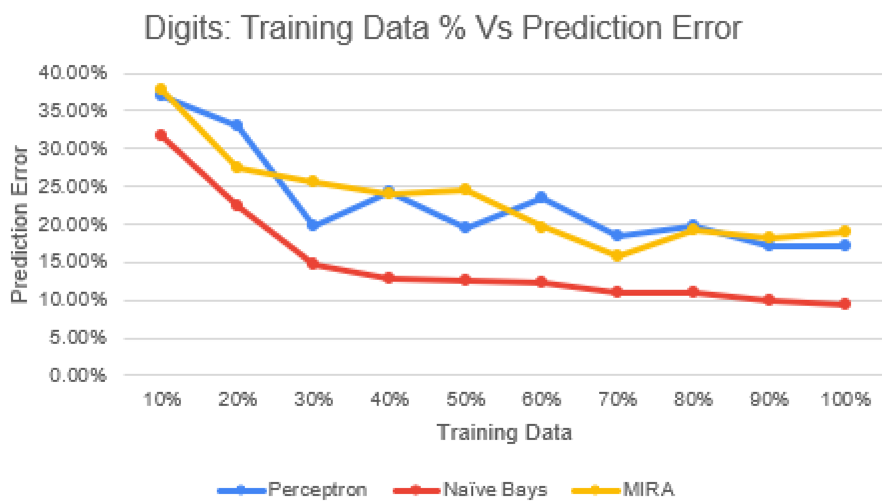
Accuracy for digits and faces are plotted against run time for each algorithm. Additionally, the run time is plotted against training data.

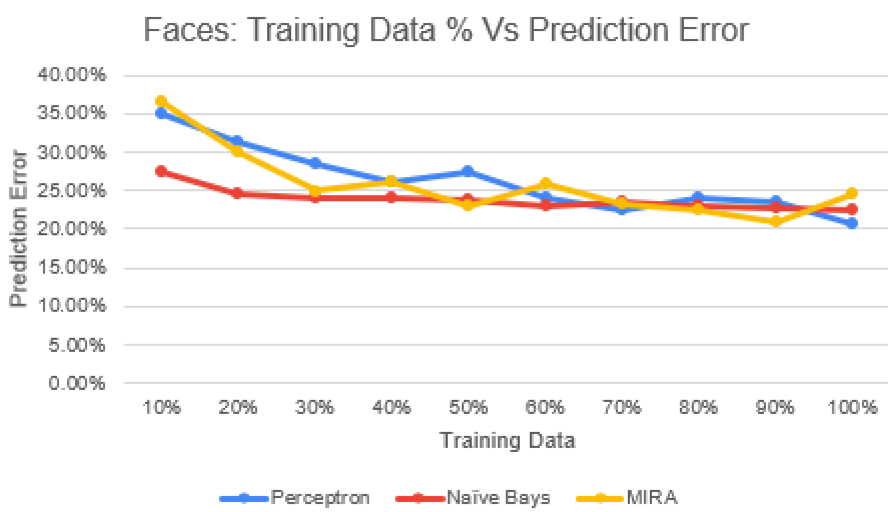

Digits: Training Data % Vs Accuracy %
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
Accuracy %
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Training Data %
Perceptron
Naïve Bays
MIRA

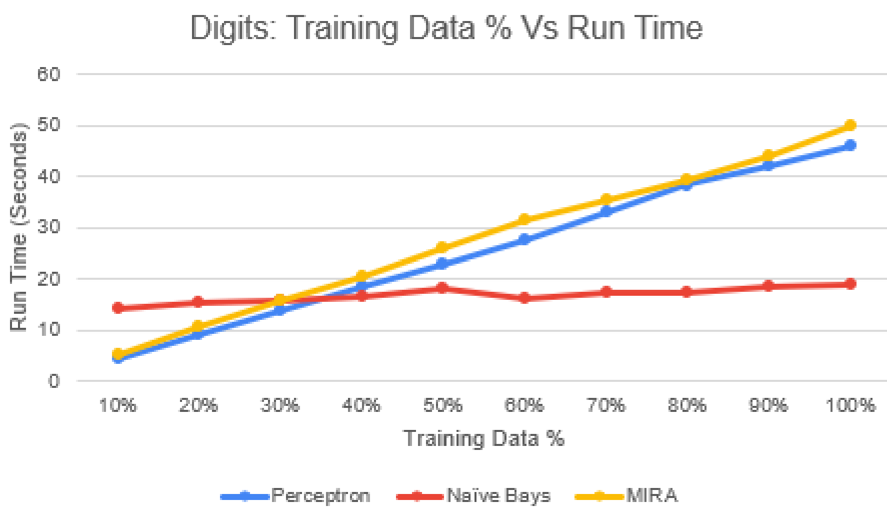

Faces: Training Data % vs Accuracy %
100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
Accuracy %
10
20
30
40
50
60
70
80
90
100
Training Data %
Perceptron
Naïve Bays
MIRA

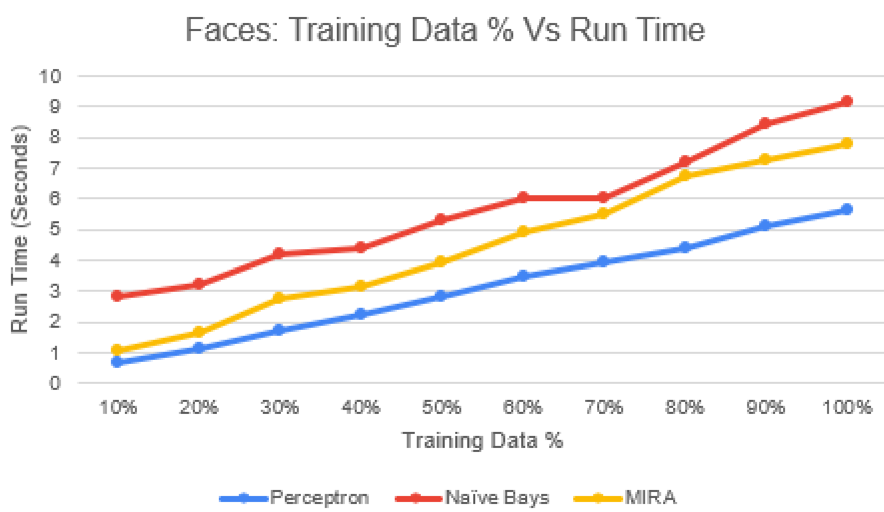

Digits: Training Data % Vs Standard Deviation
16
14
12
10
8
6
4
2
0
Standard Deviation
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Training Data %
Perceptron
Naïve Bays
MIRA


Faces: Training Data % Vs Standard Deviation
7
6
5
4
3
2
1
0
Standard Deviation
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Training Data %
Perceptron
Naïve Bays
MIRA


Digits: Training Data % Vs Prediction Error
40.00%
35.00%
30.00%
25.00%
20.00%
15.00%
10.00%
5.00%
0.00%
Prediction Error
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Training Data
Perceptron
Naïve Bays
MIRA


Faces: Training Data % Vs Prediction Error
40.00%
35.00%
30.00%
25.00%
20.00%
15.00%
10.00%
5.00%
0.00%
Prediction Error
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Training Data
Perceptron
Naïve Bays
MIRA





Analysis

Accuracy

When considering digits, Perceptron has better accuracy compared to Naive Bays and Mira. With less data, Naive Bayes is best.
For faces, Naive Bays has the best accuracy among the three. With increased training data, accuracy also increases.

Run Time

For data, we see that Naive Bays has the best run time while MIRA has the worst.
For faces, we saw a very similar slop between all three algorithms, but Perceptron gave the best run time overall.