Rahul Datta

CS 440/ ECE 448 Assignment 3: Naive Bayes Classification

Overview

This MP asked to classify images using Naive Bayes Classification. The provided training data was used to build a classifier for each class. The classifiers were then tested on the provided testing data. The algorithms were written in C++ because its a powerful object-oriented language which I am most familiar with.

Section 1.1 Digit Classification

Section 1.1 asked for handwritten digits to be classified. The algorithm first parsed the training data and created a classifier for each digit which resulted in 10 classes. The data was parsed pixel by pixel and for each digit the probability of pixel (x,y) was calculated. If a pixel is filled the probability of that pixel being on for that respective digit is increase. If not the probability is decreased. After training, the test data was parsed and tested. To test digits the digit was parsed pixel by pixel. For each pixel the log probability of that pixel being filled was calculated for each digit type. After doing this for each pixel and summing up the log probability for each pixel for each class, the class with the largest log probability was chosen. This method worked with **76.9 % accuracy.** The following are the classification rate for each digit and the corresponding confusion matrix (digits in red show the pair with the highest confusion rates).

Classification Rate for Each Digit

Digit	Classification Rate
0	0.84
1	0.96
2	0.78
3	0.79
4	0.77
5	0.67
6	0.75
7	0.73
8	0.59
9	0.80

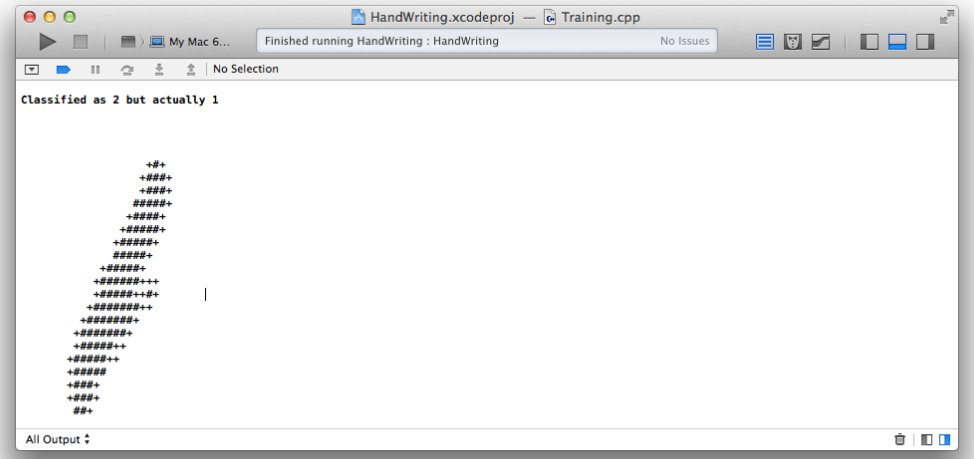
Rahul Datta

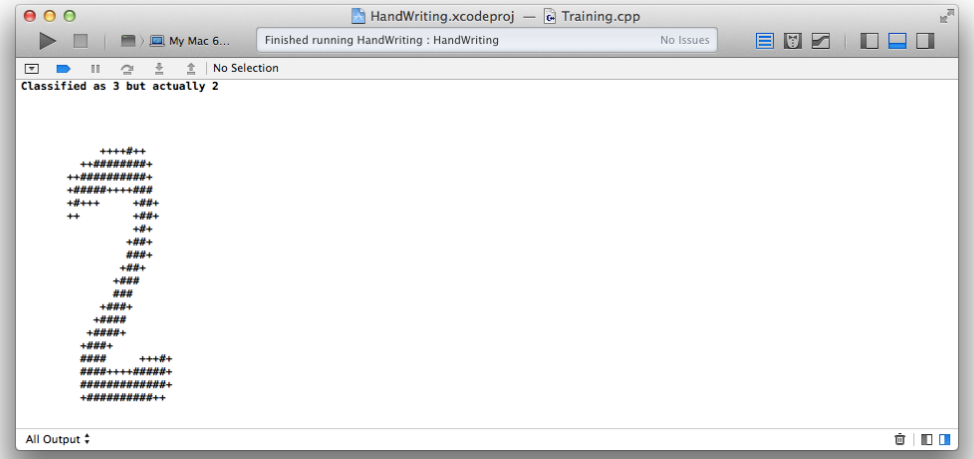
Confusion Matrix

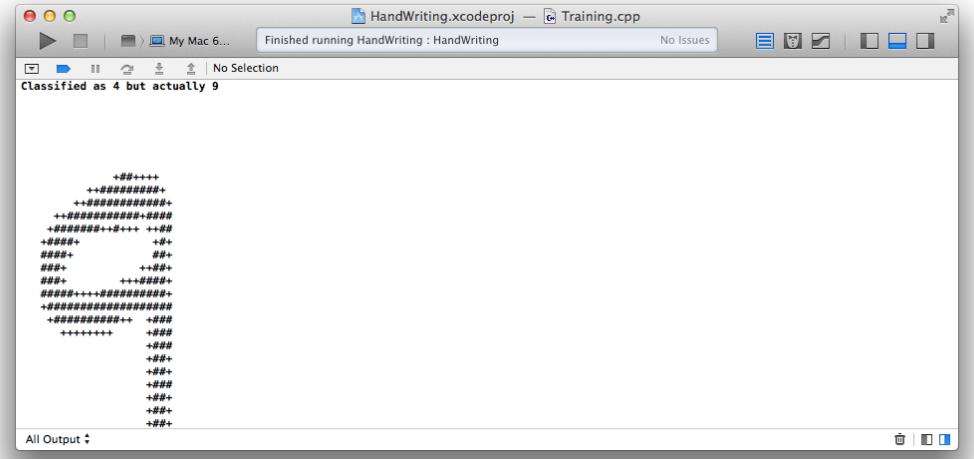
	0	1	2	3	4	5	6	7	8	9
0	0.84	0.00	0.01	0.00	0.01	0.06	0.03	0.00	0.04	0.00
1	0.00	0.96	0.01	0.01	0.00	0.02	0.01	0.00	0.00	0.00
2	0.01	0.03	0.78	0.01	0.01	0.00	0.06	0.01	0.05	0.02
3	0.00	0.02	0.00	0.79	0.00	0.03	0.02	0.06	0.02	0.06
4	0.00	0.01	0.00	0.00	0.77	0.00	0.03	0.01	0.02	0.17
5	0.02	0.02	0.01	0.13	0.03	0.67	0.01	0.01	0.02	0.07
6	0.01	0.05	0.04	0.00	0.04	0.08	0.75	0.00	0.02	0.00
7	0.00	0.06	0.03	0.00	0.03	0.00	0.00	0.73	0.03	0.13
8	0.01	0.01	0.03	0.14	0.02	0.08	0.00	0.01	0.59	0.12
9	0.01	0.01	0.03	0.03	0.09	0.02	0.00	0.02	0.01	0.80

After experimenting with different k values ranging from 1 to 50 it was determined that as the k value increase the classification accuracy decreased. Thus there was an inverse relationship between the k value and accuracy. Another interesting observation was that there was no difference between the maximum a posteriori (MAP) classification and maximum likelihood (ML). This is most likely due to the fact that there was a roughly even distribution of digits in the training data set.

Below are some interesting example of incorrect classifications made by the Naive Bayes Classification algorithm. The first example was classified as 2 but was actually a 1. The second was classified as 3 but was actually a 2 and the final image was classified as 4 but is actually a 9. What is interesting about these examples are that each of the digits do not look too different from the digits that had the highest classification probability. NOTE: It appears that some of the ASCII images are distorting upon upload to compass. If this is an issue you can access the same report at the following link: https://www.dropbox.com/s/fux7y4r5roh7oc1/Datta_Rahul_Assignment3.pdf


HandWriting.xcodeproj — Training.cpp
My Mac 6...
Finished running HandWriting : HandWriting
No Issues
No Selection
Classified as 2 but actually 1
All Output

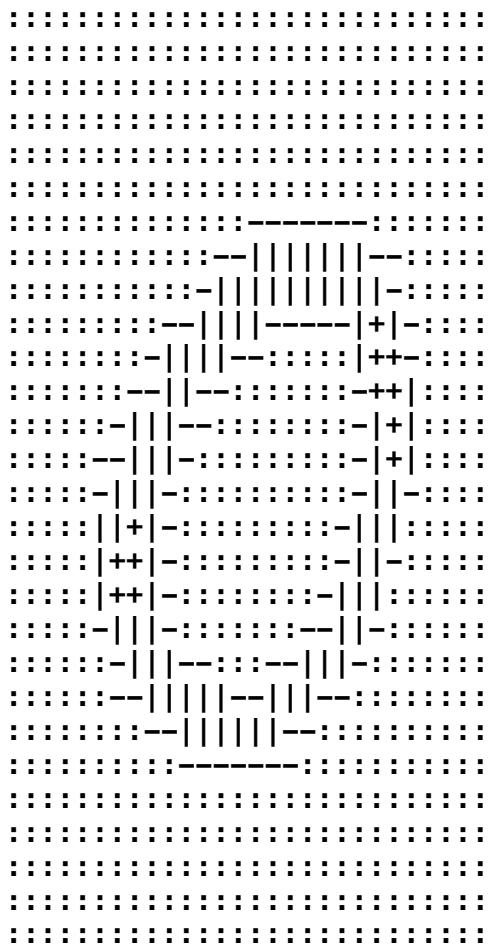

HandWriting.xcodeproj — Training.cpp
My Mac 6...
Finished running HandWriting : HandWriting
No Issues
No Selection
Classified as 3 but actually 2
All Output


HandWriting.xcodeproj — Training.cpp
My Mac 6...
Finished running HandWriting : HandWriting
No Issues
No Selection
Classified as 4 but actually 9
All Output

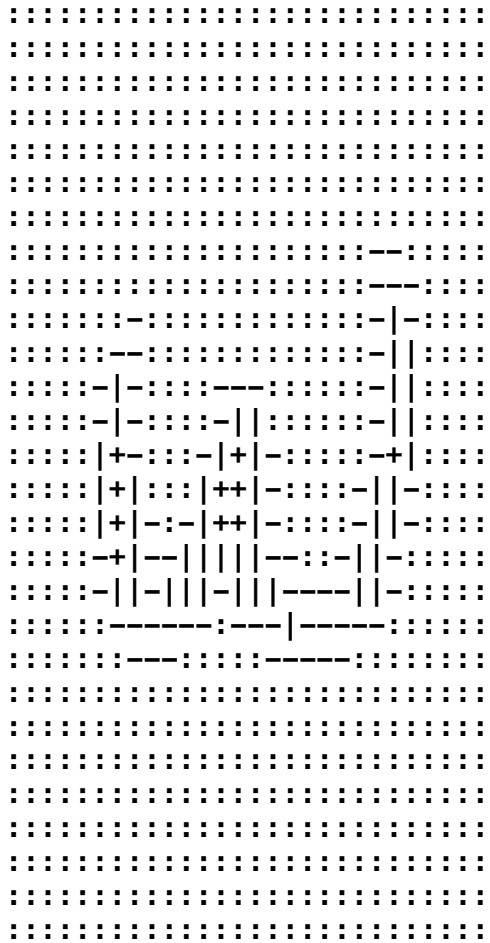
Rahul Datta

The following are the feature likelihood maps for the digits (that are not the same) that result in the highest confusion rate. 0 is included because it provides a base case and is easy to understand. The key used to create these images is as follows. A ":" was used if the probability was less than .25 or the log probability was less than -0.6025. A "-" was used if the probability was greater than .25 and less than .5 or greater than the log probability of -0.6025 and less than -0.30103. A "l" was used if the probability was greater than .5 and less than .75 or the log probability was greater than -0.30103 and less than -0.125. And finally a "+" was used to denote the probability between .75 and 1 or log probability between -0.125 and 0.

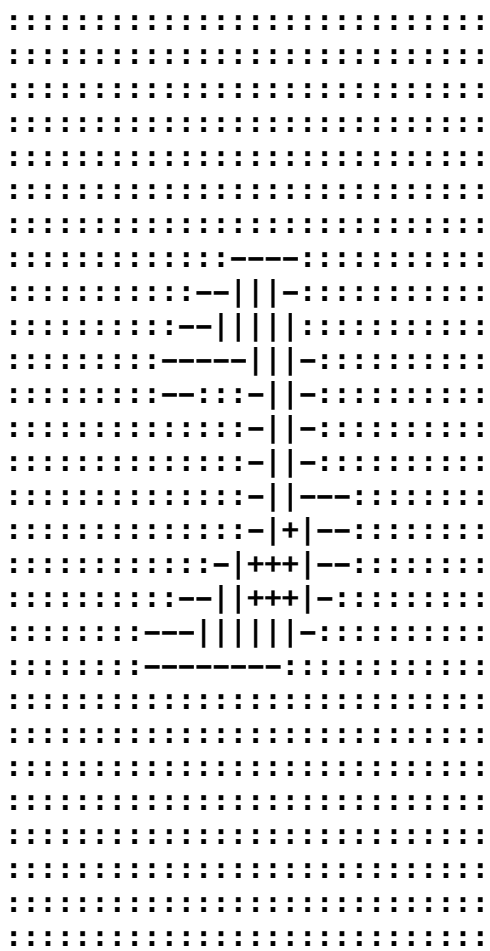
Zero



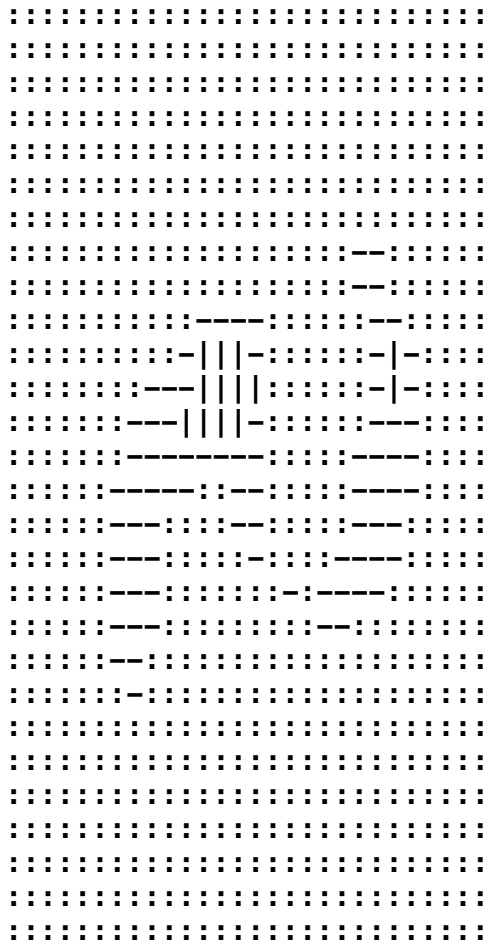
Three



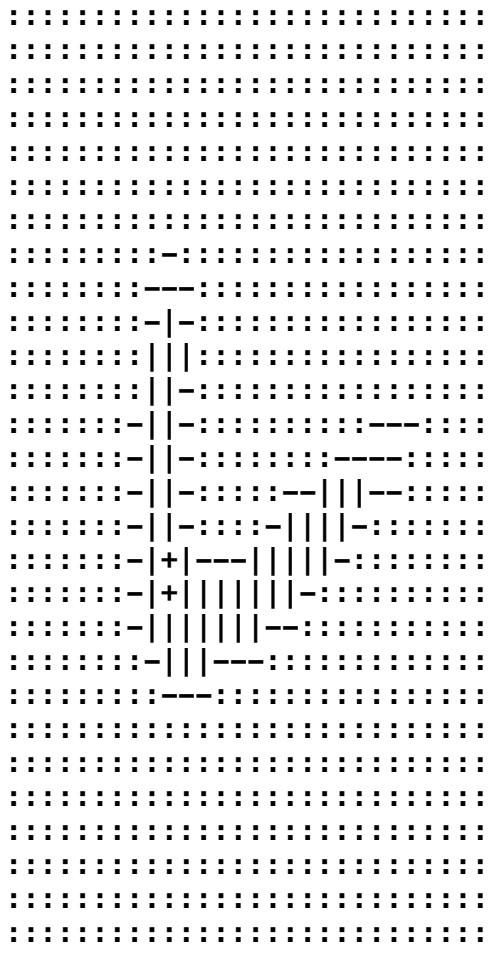
Four



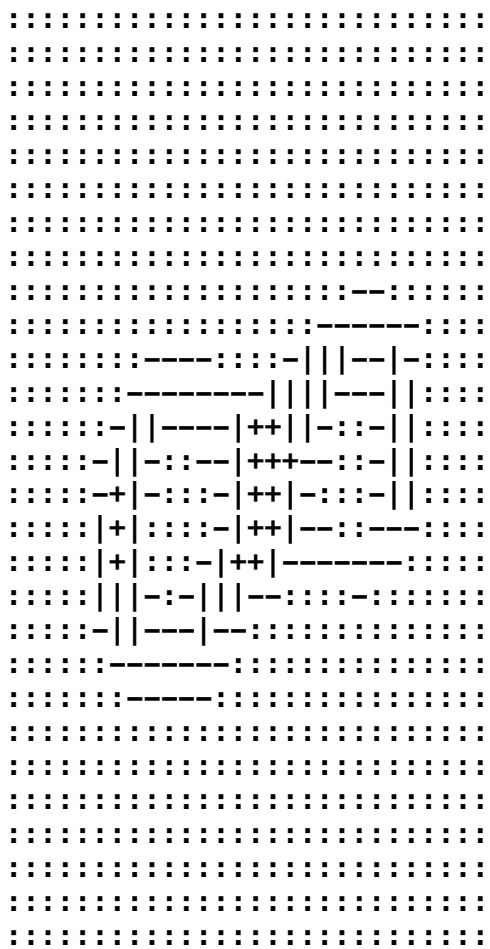
Five



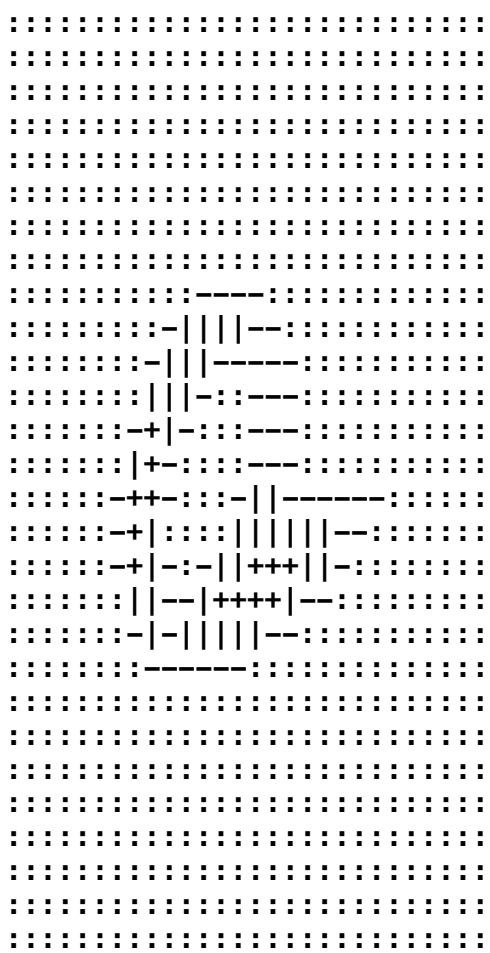
Seven



Eight

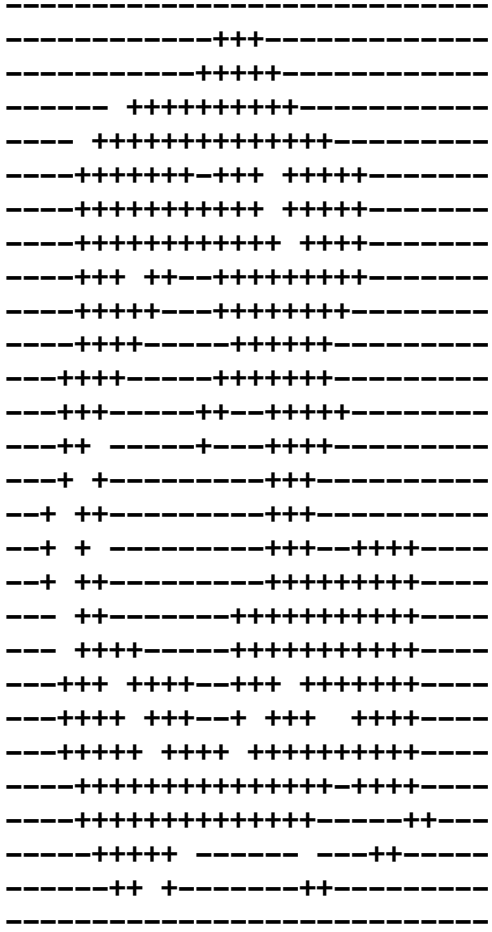


Nine

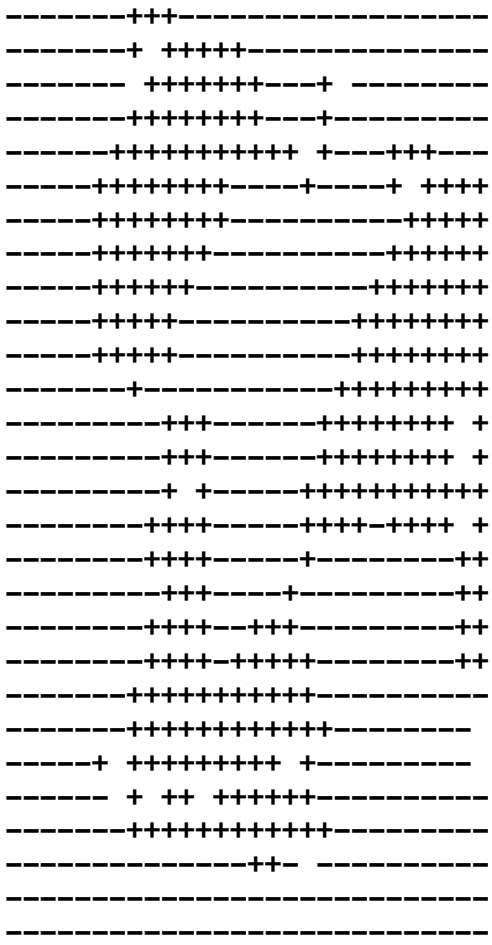


The following are the odds ratio for the four pairs of digits identified (in red) in the above confusion matrix. The following ASCII key was used for the odds ratio map. If the ratio was negative it was the pixel was assigned a '-' character. If the ratio was within 0.1 of 1 it was assigned a ' ' character and other was it was assigned a '+' to denote it was positive.

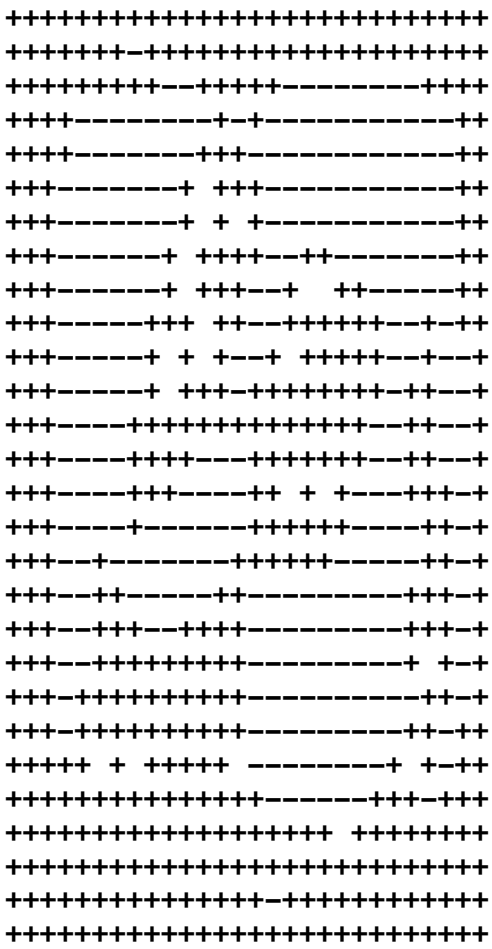
Odd ratio for 4,9



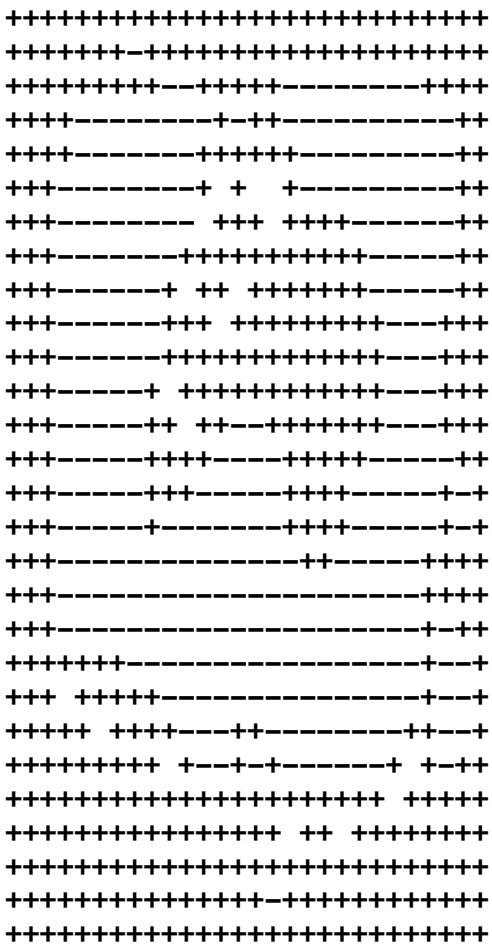
Odd ratio for 7,9



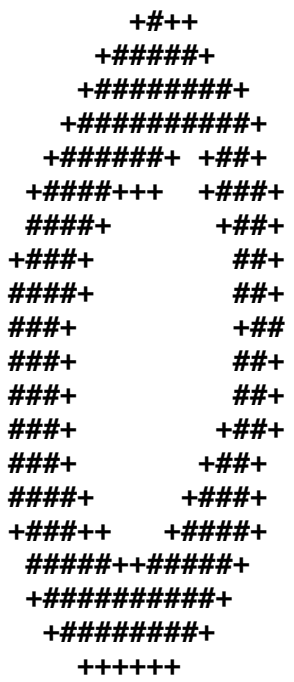
Odd ratio for 8,3

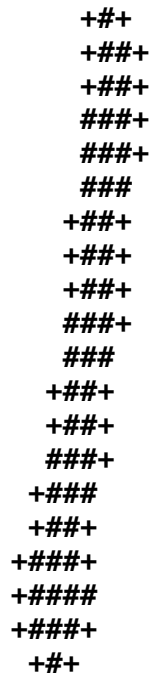


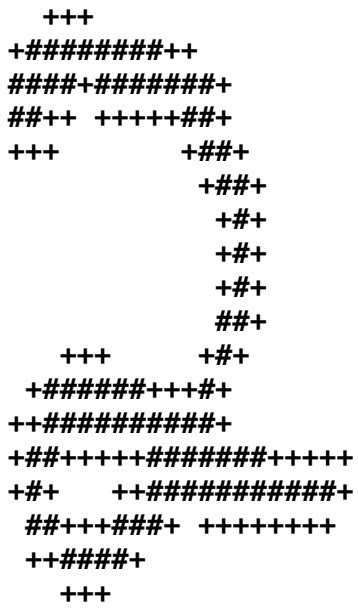
Odd ratio for 5,3

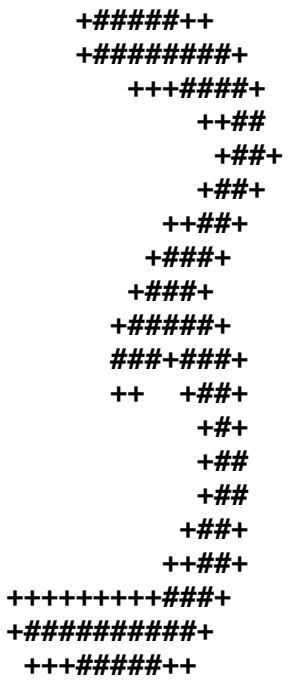


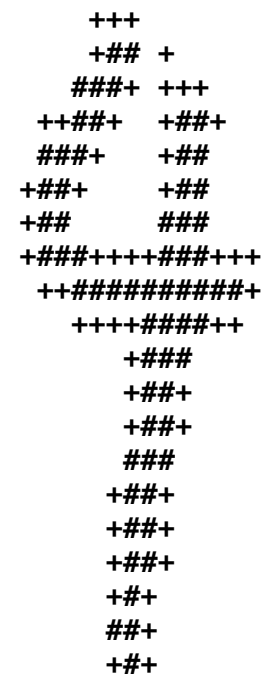
The following were the digits with the highest posterior probability

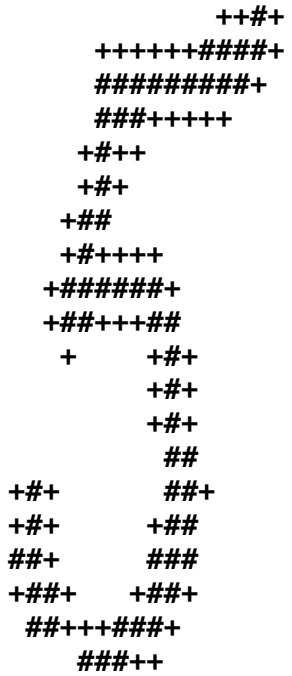


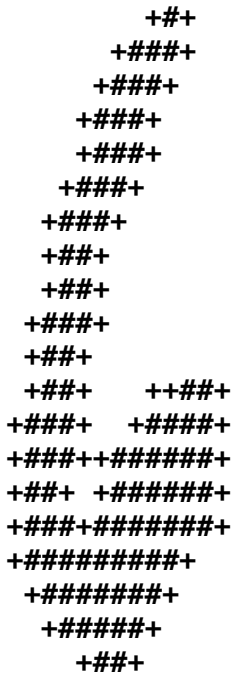


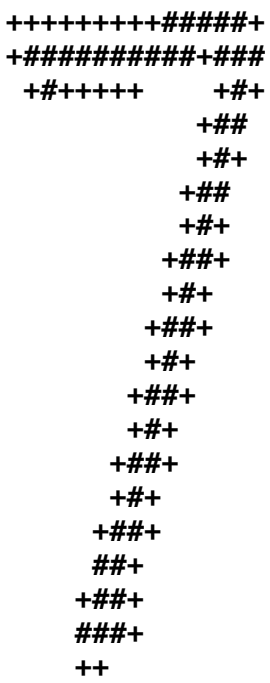


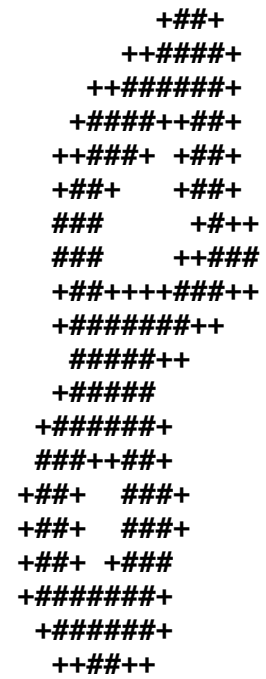


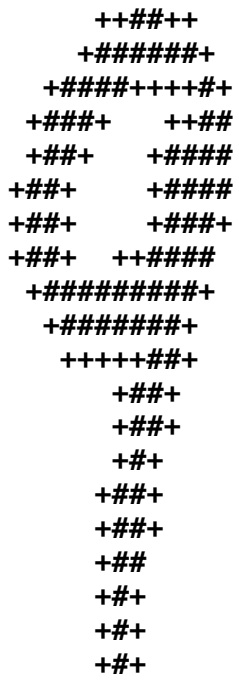












Section 1.2 Face Classification

Section 1.2 was similar to section 1.1 as it asked to classify whether an image was a face or not. For this part of the assignment the code that was used in section 1.1 was used in 1.2 with one small modification. Instead of there being ten classes, one for each digit, there were only two classes, face or non-face. With a reduced number of classes a higher accuracy was achieved. After testing the **an accuracy of 90.66%** was achieved. The following is the classification rate for categorizing faces and the corresponding confusion matrix. Again with the Lapacian smoothing the K value was 1 as it increased the accuracy the most.

Classification Rate for Faces

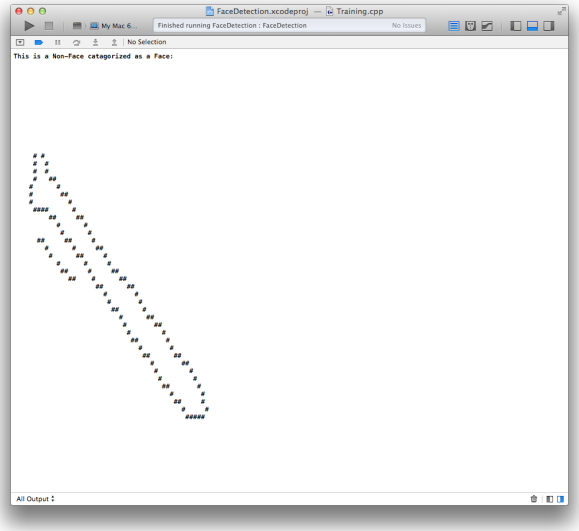
Class	Classification Rate
Non-Face	**0.88**
Face	**0.93**

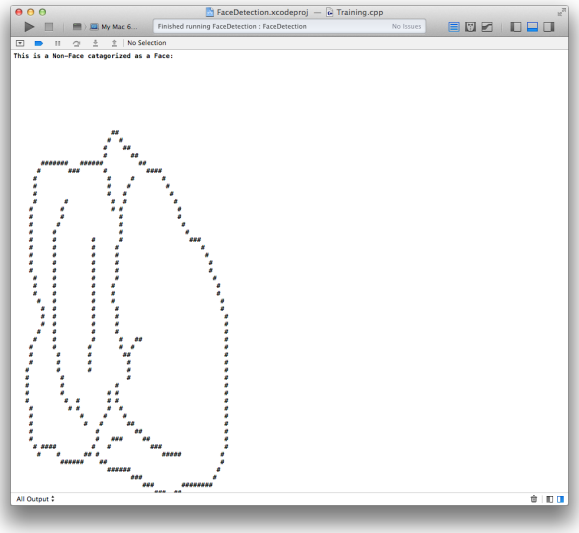
Confusion Matrix for Faces

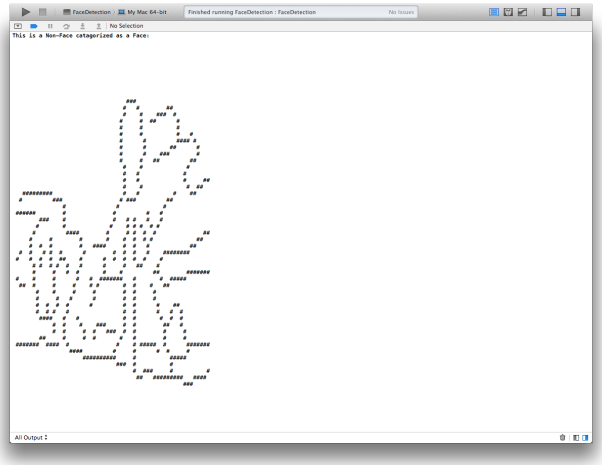
	Non-Face	Face
Non-Face	**0.88**	**0.12**
Face	**0.07**	**0.93**

Some interesting examples of false positives are below. The first images is rather curious because it does not resemble a face at all. There almost no facial features in the image that a human would be able to detect. The second image is improperly classified as a face most likely due to the fact that the image shares a lot of features with faces. With a larger training set this non-face may have been classified as a non-face instead of a face. The last false positive is interesting because it seems to have pixels on where a face would normally have pixels on, but is still not a face. Thus this non-face seems to be tricking the algorithm by have a lot of the same features of a face.

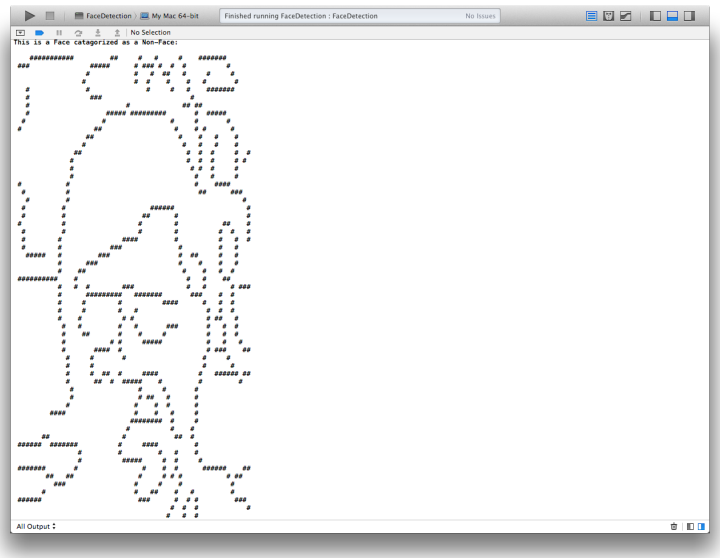
Rahul Datta

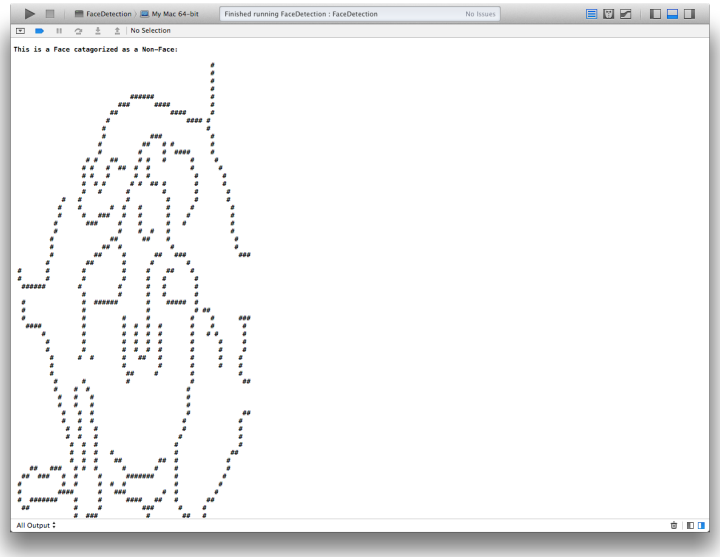

FaceDetection.xcodeproj — Training.cpp
Finished running FaceDetection : FaceDetection
No Selection
This is a Non-Face catagorized as a Face:
All Output


FaceDetection.xcodeproj — Training.cpp
Finished running FaceDetection : FaceDetection
No Selection
This is a Non-Face catagorized as a Face:
All Output


Finished running FaceDetection : FaceDetection
No Selection
This is a Non-Face catagorized as a Face:
All Output

Some interesting examples of false negatives can be seen below. The first image is interesting because even to me I had a hard time determining if I was looking at a face or garbage. After further inspection I realized it was a face but it was still difficult for me to figure out. The second image is interesting because it for the most part it seems to be categorized as a face, but there seems to be some pixels in the upper right hand corner of the image that seem to be off and thus categorizing the image as a non-face instead of a face.





Below is the odds ratio for the face data with the non-face log probabilities divide by the face log probabilities. The following ASCII key was used for the odds ratio map. If the ratio was negative

it was the pixel was assigned a ‘-‘ character. If the ratio was within 0.1 of 1 it was assigned a ‘ ‘ character and other was it was assigned a ‘+’ to denote it was positive