INTRODUCTION

A subset of MNIST dataset containing handwritten digits of 7 and 8 is given. The aim of the project is to train two different image classifiers to classify these two digits. Naive Bayes Classifier and Logistic Regression Classifier are used in this scenario. Both of them are trained with two features. The features that are extracted from both the images are assumed to be independent. The features used are:

1. Mean value of all the pixel values in the image
2. Standard deviation of all the pixel values in the image

Random samples of dataset are as shown below:

The programming language used here is Python 3.x and Scipy and Numpy are used as supporting libraries.

Naive Bayes Classifier

This classifier hinges on the Bayes theorem which has the following formula:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

- P(C|X) indicates the posterior probability that the class is C given the data X.
- P(X|C) indicates the likelihood that data X belongs to class C
- P(C) indicates the class prior probability
- P(X) indicates the predictor prior probability

The class chosen using this rule is simply the one that yields the highest probability for that data.

$$c^* = argmax_c P(c|x) = argmax_c \frac{P(x|c)P(c)}{P(x)}$$

Since P(X) is constant for all values of C in P(C|X), we ignore P(X) when we take the argmax. Hence the equation becomes:

$$c^* = argmax_c P(x|c)P(c)$$

Since log function is a continuously increasing function, we can transform the above equation into:

$$c^* = argmax_c (logP(x|c) + logP(c))$$

P(C) can be calculated from the dataset using the following formula.
P(c) = count(number of times images of the digit c appear) / count(total number of images)

For modeling probability densities, we can use the multivariate normal distribution. Since, we are assuming the two features extracted to be independent of each other, we can calculate the joint probability of a class by multiplying the product of two individual probabilities. This brings us to our final equation.

$$P(x|c) = \prod_{i=1}^{2} \frac{1}{\sqrt{2\pi\sigma_i^2}} exp\left(-\frac{1}{2}\frac{(x_i-\mu_i)^2}{\sigma_i^2}\right)$$

Results:

P(7) = 0.517084846484
P(8) = 0.482915153516
μ (7) = [0.1145277, 0.28755657]
σ2 (7) = [0.00093834, 0.00145932]
μ (8) = [0.15015598, 0.32047584]
σ2 (8) = [0.00149247, 0.00159681]
Accuracy of Naive Bayes classifier: 69.530470 %
Accuracy of Naive Bayes classifier for 7: 75.972763 %
Accuracy of Naive Bayes classifier for 8: 62.731006 %

Logistic Regression Classifier

Logistic regression classifier is a classifier that predicts the probability of an outcome that has only has two values. This classifier produces a logistic curve which is limited to values between 0 and 1. This is done with the help of a function called the logistic function or the sigmoid Function.

$$z = \Theta^T x$$
$$sigmoid(z) = \frac{1}{1 + e^{-(z)}}$$

Θ is the weight matrix.

This function needs to be optimised by the Maximum Likelihood Estimation(MLE) approach. Maximizing the likelihood can be done by gradient descent.

Log likelihood equation of the logistic regression function is shown below:

$$ll = y \cdot z - log(1 + e^z)$$

Gradient of the log likelihood equation is the derivative of the above equation.

$$\bigtriangledown ll = X^T(y - g(X\Theta))$$

The weights are then updated by adding the derivative times the learning rate at each iteration.

$$\Theta := \Theta + \alpha \cdot \frac{\delta loss(\Theta)}{\delta \Theta}$$

Results

Number of iterations: 100000
Learning rate: 0.003
Weights: [81.90347798 853.03090674 -620.23365695]
Final sigmoid equation: 1/(1+exp(-(81.90347798 + 853.03090674*X1 + -620.23365695*X2)))
Accuracy of Logistic Regression classifier: 73.026973 %
Accuracy of Logistic Regression classifier for 7: 49.416342 %
Accuracy of Logistic Regression classifier for 8: 100.000000 %