BLG 454E Learning From Data (Spring 2018)

Homework 2

Onat Sahin - 150150129

1 Question 1

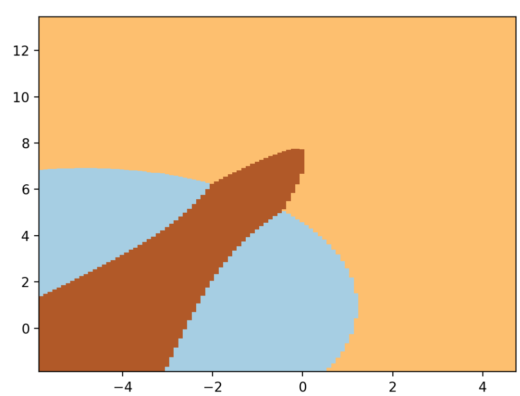
The code for this question can be found in q1.py .

a) Formula for the discriminant function g(x):

$$g_i(x) = -(1/2)(x - \mu_i)^t \Sigma_i^{-1} (x - \mu_i) - (1/2) \ln |\Sigma_i| + \ln P(c_i)$$

This is a more convenient form of the function given in the question

b)Plot for the calculated decision boundaries:



c)The test accuracy is 79%. This means 158 cases out of 200 are predicted correctly.

2 Question 2

The code for this question can be found in q2.py .

a) A sample output of my code is given below.

```
Learning rate:  0.1
Accuracy:  96.0 %

Confusion Matrix:
[[50  0  0]
 [ 0 48  2]
 [ 0  4 46]]
```

For this sample, I used 0.1 as learning rate with 300 iterations. The accuracy found here is 96%. To achieve better training, I randomly shuffle the given data to change their order in addition to initializing weights randomly. Because of this, the results differ slightly between different runs of the program. However, these differences are very minor. In the confusion matrix, rows represent the true class of the data while columns represent the predicted class. The first row and column represent iris setosa, while the second row and column represent iris versicolor, and the third ones represent iris virginica. From the confusion matrix given in the sample it can be seen that the algorithm manages to clasify iris setosa very accurately, but sometimes confuses iris versicolor and iris virginica. This holds true in multiple runs of the program. Iris versicolor and iris virginica are always most confused with each other.

b) I ran my program with the given learning rates using 500 iterations and 300 iterations. Because I randomly shuffle the given data in the program when the data is read and weights are initialized randomly, the results I get tend to be slightly different each time the program is run. Therefore, I had to run my program multiple times for each iteration and learning rate. With 500 iterations, accuracies were really similar between different learning rates with 0.01 giving slightly better results. With 300 iterations, learning rate 0.1 gave the best results, even better than 0.01 with 500 iterations. However, other learning rates became less stable, sometimes giving high and sometimes low accuracies. But in the end, like 500 iterations, accuracies did not have big differences. I think with these amounts of iterations, the algorithm manages to converge even with bigger and smaller learning rates.