BLG 454E Learning From Data (Spring 2018)

Homework II

1 Question 1

This question looks similar to that we had in homework 1. This time we have 2 features but 3 classes. So multivariate analysis we will predict classes in the test set. I used three functions in this question but script
automatically train the model after execution. So before executing the function in the main it trains the model.
After that plot_the_result function works and it basically predicts all the values from min of the test data to max of test data then print it with matplotlib.

1.1 Part a

The conditional probability of a class is given in the homework as :

$$P(x \mid C_i) = \frac{1}{\sqrt{(2\pi)^d \mid \sum_i \mid}} \exp\left(-\frac{1}{2}(x-\mu)^T (\sum_i)^{-1}(x-\mu)\right)$$

And we have class probability for every class from the train dataset which i will write as:

$$P(C_i) = \frac{number of c_i}{number of all data}$$

Also discriminant function of any x is:

$$g_i(x) = \ln P(x \mid C_i) + \ln P(C_i)$$

If we plug our two probabilities on the above we get:

$$g_i(x) = -\frac{1}{2}(x-\mu)^T (\sum_i)^{-1}(x-\mu) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln \mid \sum_i \mid + \ln P(C_i)$$

And eliminating constants give us final cost function to use in prediction which is:

$$g_i(x) = -\frac{1}{2}(x-\mu)^T (\sum_i)^{-1}(x-\mu) - \frac{1}{2}\ln \mid \sum_i \mid + \ln P(C_i)$$

As you can see i need covariance matrix in this code and for that i need mean vector. I calculated them with :

```
for i in traindata:
    if i[2] == 0:
        meanVector[0][0]+=i[0]
        meanVector[0][1]+=i[1]
        class0data = np.vstack([class0data,i]) if len(class0data) else i
        classCounter[0]+=1
    if i[2] == 1:
        meanVector[1][0]+=i[0]
        meanVector[1][1]+=i[1]
        class1data = np.vstack([class1data,i]) if len(class1data) else i
        classCounter[1]+=1
    if i[2] == 2:
        meanVector[2][0]+=i[0]
        meanVector[2][1]+=i[1]
        class2data = np.vstack([class2data,i]) if len(class2data) else i
        classCounter[2]+=1


for i in range(3):
    meanVector[i]=np.divide(meanVector[i],classCounter[i]) #We got mean
        vector for all classes
```

By getting meanVectors of each class in meanVector matrix i calculated covarianceMatrix as follows. It is redundant to use 3 different variable for each matrix but i started to write the code like this and couldn't find a time to change so there are 3 different covariance matrix with 3 different variable.

```
class0covarianceMatrix = np.zeros((2,2))
for i in range(2):
    for j in range(2):
        for k in range(classCounter[0]):
            class0covarianceMatrix[i][j]+=(class0data[k][i]-meanVector
                [0][i])*(class0data[k][j]-meanVector[0][j])
        class0covarianceMatrix[i][j] = class0covarianceMatrix[i][j]/(
            classCounter[0]-1)
```

1.2 Part b

Here you can see my plot that shows class boundaries in Figure 1. I didn't put the test results on the plot since homework pdf doesn't want us to do.

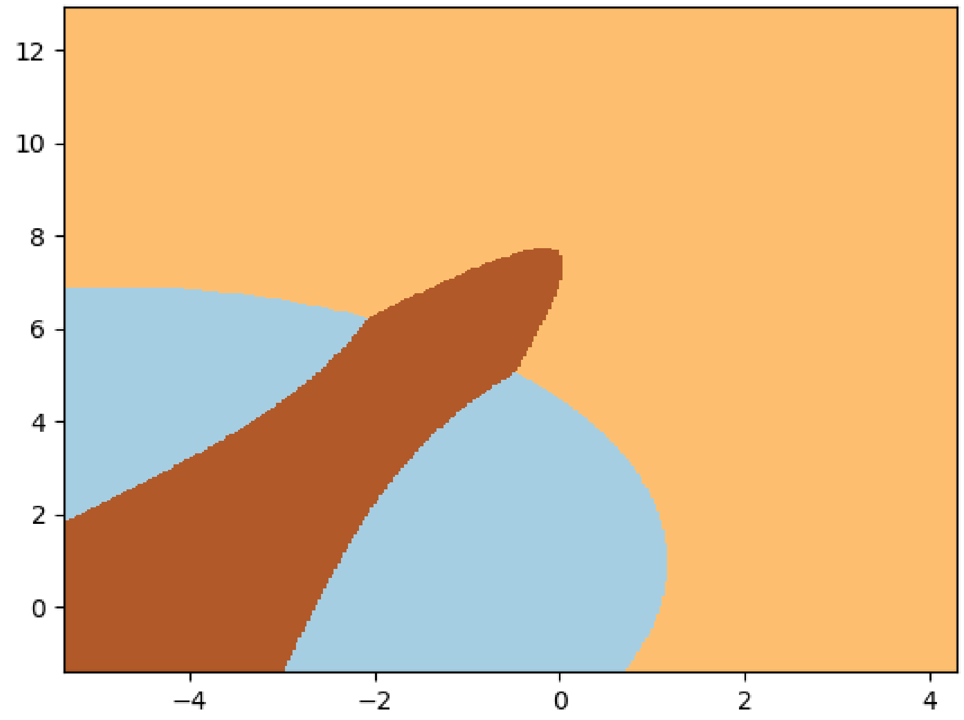


Figure 1: Class boundaries.

1.3 Part c

I get 79.5% accuracy in the test data. You can see the result by running python code as *pythonquestion*1*.py* Also script will give the plot that i put on the above with accuracy in the testdataset. It takes a while until calculate the boundaries. At the end of the testing you can see both plot and accuracy.

2 Question 2

In this question since pseudo code convention is little confusing it took some time. I used the pseudo code that given in the pdf. However there was a overflow issue in the softmax function so i used little different mathematical approach for softmax function which guarantee there is no overflow. I also used it as vectorized so there is no for loop in softmax function. In machine learning we always try to use vectorized approach and i used in here as well.

2.1 Part a

I executed the python script that i wrote and you can see my confusion matrix for 10 fold validation. I want to remind you that learning rate was 0.1 in this run. As we can see that class 0 predicted perfectly but there are some predictions that real class is 2 but predicted as 1. Also there are some predictions that real class is 1 but predicted as 2. So i can say that class 1 (Iris-versicolor) and class 2 (Iris-virginica) is confused with each other.

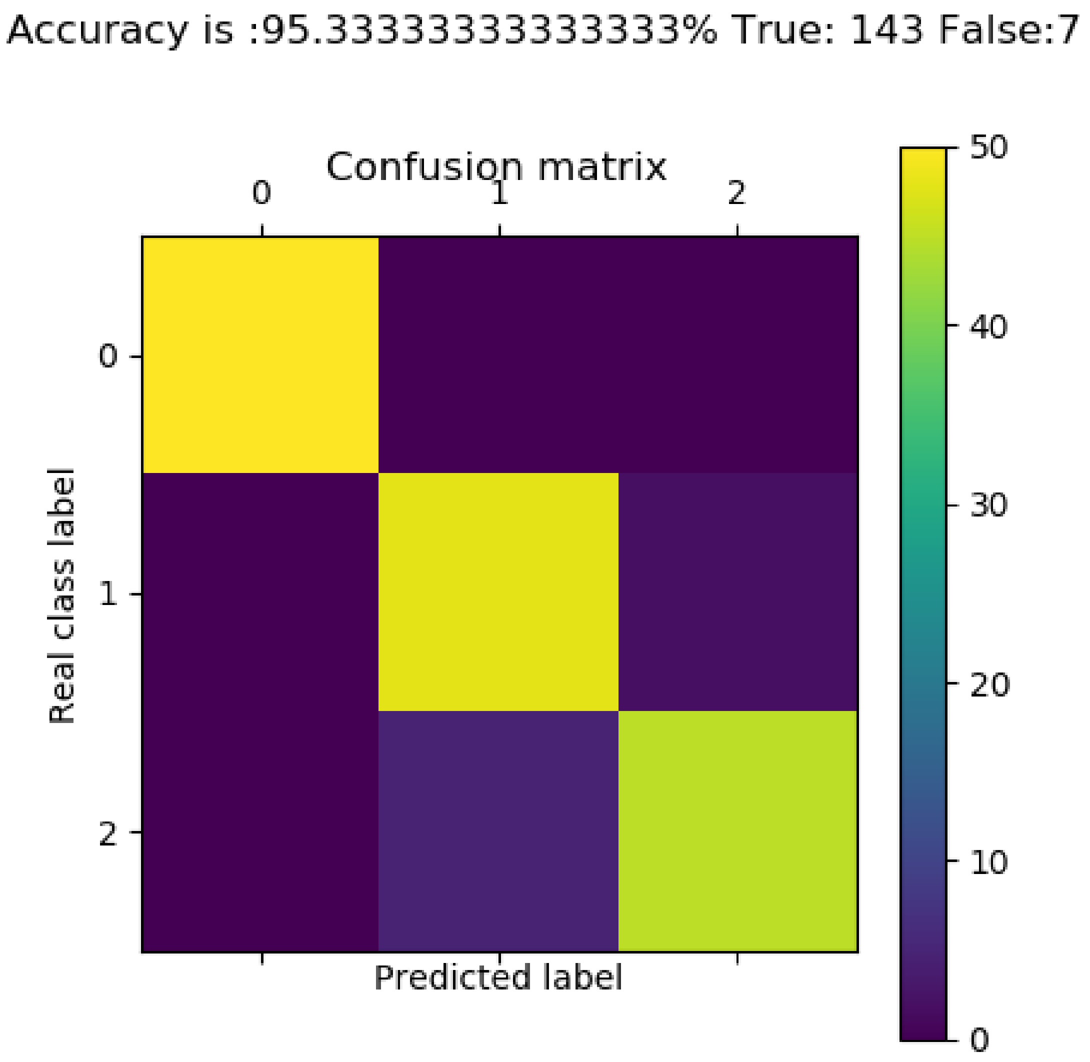
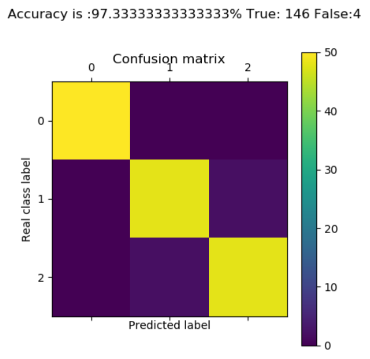


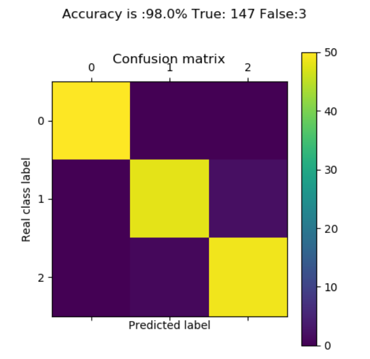
Figure 2: Confusion matrix.

2.2 Part b

I am putting below my confusion matrices and their accuracy for different rates. Since i am shuffling the train set and using random w values for each train. There was no big significance in accuracy

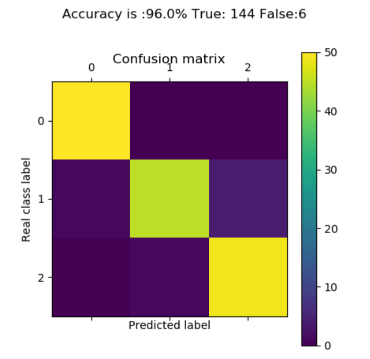


(a) Learning rate=10

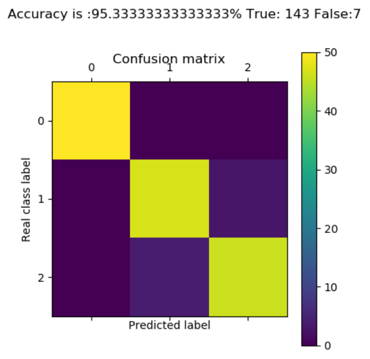


(b) Learning rate=1

Figure 3: Confusion matrix and accuracy with different learning rates



(a) Learning rate=0.1



(b) Learning rate=0.01

Figure 4: Confusion matrix and accuracy with different learning rates

but number of iteration to converge is differ. If stepsize(learning rate) is smaller than it will need more iteration to find minimum value. Also there is a risk that if step size is so small then it can stuck in the local minimum and cannot find global minimum value. If learning rate is big than iteration that we need will be less but it can miss the global minimum this time. So if it always miss the global minimum then it wouldn't find the minimum value since it always step with big values. You can see the different learning rates in the above.