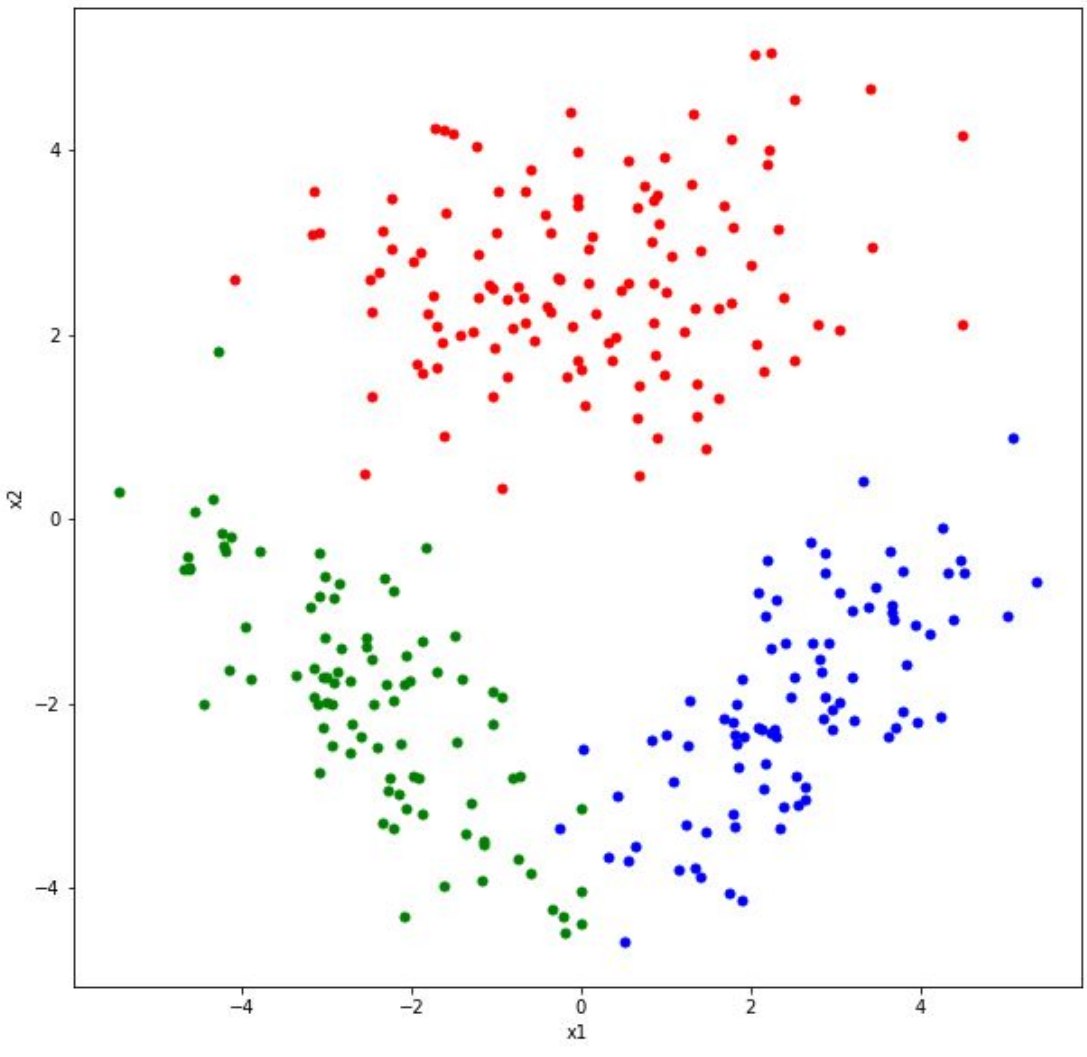
İrem Şahin

ENGR421

Mehmet Gönen

HW1-Report

For this project, I have written my code in a Spyder .py file. I have started my code by importing necessary libraries, selecting a random seed and writing our parameters down which were given in the pdf file. After this I have generated random samples by using numpy's random.multivariate_normal built-in method. I have chosen to first save these points to a csv and then extract them to the .py file. I have assigned my data points to X and their corresponding classes to y_truth, my data generation was finished successfully. My generated data points can be seen below:



I have continued by taking K(# of classes) and N(# of samples) values for future use. Next step was to estimate the sample mean, covariance and class probabilities of our generated data. Again, by getting help from the numpy library, these were trivial to calculate. My estimations for seed number 421 were as below:

```
Sample means: [array([0.04453807, 2.61225132]), array([-2.48491072, -1.94817984]),
array([ 2.54954733, -1.98880993])]

Sample covariances: [array([[2.81619315, 0.22436505],
       [0.22436505, 1.00404695]]), array([[ 1.53076998, -1.18878269],
       [-1.18878269,  1.57281213]]), array([[1.39930016, 0.92292252],
       [0.92292252, 1.24783102]])]

Class probabilities: [0.4, 0.3, 0.3]
```

...which were pretty close to the parameters we have entered at the beginning.

Next, we needed to calculate each data points score for each class and assign the data point to the class of their highest score. For this, I implemented the gc(x) function we have seen in our multivariate classification class. I have calculated the score of the data points for each class, and added them as 1x3 arrays to the Y_predicted array. In the end of gc(x) calculations I had a 1x300 array which had 1x3 arrays for each item.

After getting the Y_predicted correctly, I have used numpy's argmax function. By taking the max element's index for each data point and adding them 1 (because it is 0-indexed), I have extracted the class estimations for each point. By crosstabing it with our y_truth from the beginning, I was able to print a confusion matrix to see which many points were misclassified and where they were classified to. My confusion matrix was as below. 1 point from class 2 and 1 point from class 3 were misclassified.

y_truth	1	2	3
y_pred			
1	120	1	0
2	0	89	1
3	0	0	89

Last step was to visualize our results. First I have created grids to plot our data on. Then, by using the W, wc and wco's that we have calculated in gc(x) I have calculated the decision boundaries. I have used a quadratic formula(as seen on the code in .py file) to draw curved lines rather than linear ones because curved ones are a bit more accurate, but I have not overfitted the data either. By using contour I have drawn the lines, and by using contourf I have filled the regions with appropriate colours. I have also marked the misclassified points clearly. Final plot for seed 421 was as below: