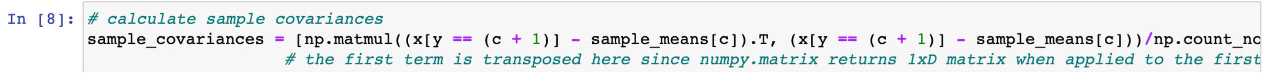
ENGR421

HW1

Doğa Demirtürk
68859

I started this homework by investigating the lab assignments. To create sample points and top lot the data, I used the code from lab01. I interpreted them to use in multivariate case with D = 2 and K = 3. One of the hardest thing I encountered there in the code was the dimensions of the variables. I realized when working with the transposes of the vectors that the dimension representation in Python (in the libraries I used) was different from expected. So in the code, sometimes I used transpose where in the theorems it was not expected. One example of this usage is the following (it should have been (x-mean) * transpose(x-mean) by the theorem):



```
In [8]: # calculate sample covariances
        sample_covariances = [np.matmul((x[y == (c + 1)] - sample_means[c]).T, (x[y == (c + 1)] - sample_means[c]))/np.count_nc
                            # the first term is transposed here since numpy.matrix returns 1xD matrix when applied to the first
```

The fourth part was a bit more challenging. I calculated the unknown parameters using the notes from the lecture:

$$W_c = -\frac{1}{2} \hat{\Sigma}_c^{-1}$$

$$w_c = \hat{\Sigma}_c^{-1} \cdot \hat{\mu}_c$$

$$w_{co} = -\frac{1}{2} \hat{\mu}_c^T \cdot \hat{\Sigma}_c^{-1} \cdot \hat{\mu}_c - \frac{D}{2} \log(2\pi) - \frac{1}{2} \log(|\hat{\Sigma}_c|) + \log[\hat{P}(y=c)]$$

Then I calculated the confusion matrix using a function of called crosstab in Python. The last part was the most challenging one. I searched on the internet for a considerable amount of time to find how to properly draw a line and select the points that does not match with the estimated values. My lack of experience in Python may be the cause of that challenge.