Name: Thuan Tran
CSS 490 Homework 2 Part 3
Date: May 6th, 2017

REPORT

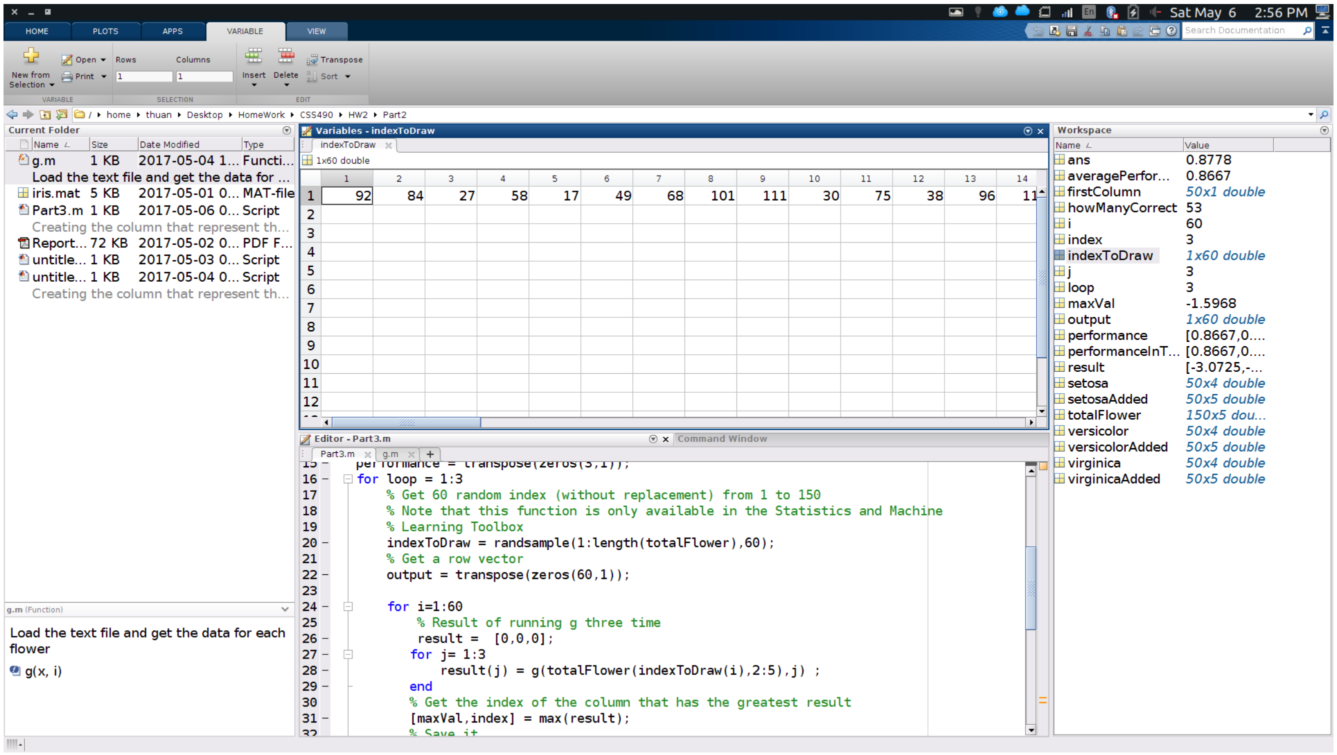
NOTE

To test out this part, include the iris.mat data set, this script and the function g.m that I have attached to the zip file. Before running this script (Part3.m), navigate the the iris data set and double click on it to load the file. When you load it, there should be 3 variables called setosa, virginica and versicolor. After that, run this script to see the performance on 3 trials and the average performance.
I also used a function called randsample that is available in the Statistics and Machine Learning Tool box to draw 60 random index with no replacement from the 150 observations of all flowers

Algorithm

Here are my steps for this assignment:

1) First I created a column vector that is all 1 (Note that I decided to choose setosa as 1, versicolor as 2 and virginica as 3). Then I append that column to the setosa flower. After that, I did the same for the other flower (only change the data in the column vector then append it to it respective flower)
2) Then I concatenate all 3 flowers into a single variables with 150 rows and 5 columns
3) Because the assignment as for 3 trials, I created a for loop that loop 3 times
4) Inside the 3 time loop, I draw a random sample of 60 numbers (index) that is from 1 to 150. Note that the function require the Statistic and Machine Learning Tool box (Illustration 1)
5) After knowing the 60 index I need to draw, I only need to run the function g(x,i) 60 times with 60 index. And each time I will also run 3 times for different I. I then save those result (the one with the largest) into a variable for comparison later on
6) After that, I only need to check how many response I got right by going through the result again and comparing them with the result of the actual data. I then performed calculation and save them into a variable that represent the performance of this trials. Then I repeat the trials for 2 more times.



	Name	Size	Date Modified	Type
	g.m	1 KB	2017-05-04 1...	Functi...
	Load the text file and get the data for ...			
	iris.mat	5 KB	2017-05-01 0...	MAT-file
	Part3.m	1 KB	2017-05-06 0...	Script
	Creating the column that represent th...			
	Report...	72 KB	2017-05-02 0...	PDF F...
	untitle...	1 KB	2017-05-03 0...	Script
	untitle...	1 KB	2017-05-04 0...	Script
	Creating the column that represent th...			

Load the text file and get the data for each flower

g(x, i)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	92	84	27	58	17	49	68	101	111	30	75	38	96	11
2														
3														
4														
5														
6														
7														
8														
9														
10														
11														
12														

```
performance = transpose(zeros(3,1));
for loop = 1:3
    % Get 60 random index (without replacement) from 1 to 150
    % Note that this function is only available in the Statistics and Machine
    % Learning Toolbox
    indexToDraw = randsample(1:length(totalFlower),60);
    % Get a row vector
    output = transpose(zeros(60,1));

    for i=1:60
       % Result of running g three time
       result =  [0,0,0];
      for j= 1:3
          result(j) = g(totalFlower(indexToDraw(i),2:5),j) ;
      end
      % Get the index of the column that has the greatest result
      [maxVal,index] = max(result);
      % Save it
```

Name	Value
ans	0.8778
averagePerfor...	0.8667
firstColumn	50x1 double
howManyCorrect	53
i	60
index	3
indexToDraw	1x60 double
j	3
loop	3
maxVal	-1.5968
output	1x60 double
performance	[0.8667,0....
performanceInT...	[0.8667,0....
result	[-3.0725,-...
setosa	50x4 double
setosaAdded	50x5 double
totalFlower	150x5 dou...
versicolor	50x4 double
versicolorAdded	50x5 double
virginica	50x4 double
virginicaAdded	50x5 double

Illustration 1: Sample output of the randsample method

TESTING APPROACH

Here is my testing approach. After I performed concatenating the flowers, I checked in MatLab by clicking on the variables to see if they are in the correct order that I concatenate and if the new variable is a 150x5 matrix

In addition, when calculating the performance. I also put a break statement there to make sure that the index(result) generated matched with the result of the flower.

Here are the details: I have a 1x60 array that contains the index that I am going to draw like 142,136,45,78,... After calling the function g(x,i) 3 times, i put the value, which is the index of the class that is optimal in the a 1x60 result array

So for that array, it now has the result of running g with 3 classes and selecting the optimal result for 60 times.

So that array will have result like 1,2,1,3,1 where the index of it correspond to the index of the 1x60 array that have the index I am going to draw. Like for the result array, at index 1, the value is 1. That is the value of running g(x,i) 3 times of row 142 in the 150x5 array where index 142 was selected first and put in the position at index 1 at the index Array.

So while calculating the performance. I need make sure that I get the correct row for the result array

I once did an error like : result(i) == totalFlower(i). While instead it should be result(i) == totalFlower(index(i)) to match the correct flower

In addition, I also tested the method multiple time in the command window where i passed in the data of the setosa, versicolor, virginica to see if g(x,i) match the class that I passed in. Most of the time, the setosa produced correct result.

ANALYSIS

For this function, I am not surprised to see that it did not generate a 100% correct rate.   One of the reason why it did not do that because it used a general covariance matrix. That assumes all classes has the same covariance matrix. While in fact, each class should use its own covariance matrix because each features correspond differently to another features in each class (Case 3)
Further more, looking at the plots of Part 0 (Illustration 2). Even though the setosa is easy to differentiate. The versicolor and virginica are not since they are clustered toward each other. Hence, it is not possible to generate a LINEAR line to classify those two using the linear discriminant function. That is because the function rely on the Mahalanobis distance from each observation x to the means. And Illustration 3 shows that the means are near to each other, hence hard to differentiate the class.
Furthermore, the plots for the distribution of flowers (Illustration 3) also suggest that it is hard to differentiate the flowers as well Since all the plots are clustered with each other. The standard deviation of the virginica also extended into the region of versicolor as well and setosa. Looking at the picture, I can say that about half of the each flower fall into the region of another flower.

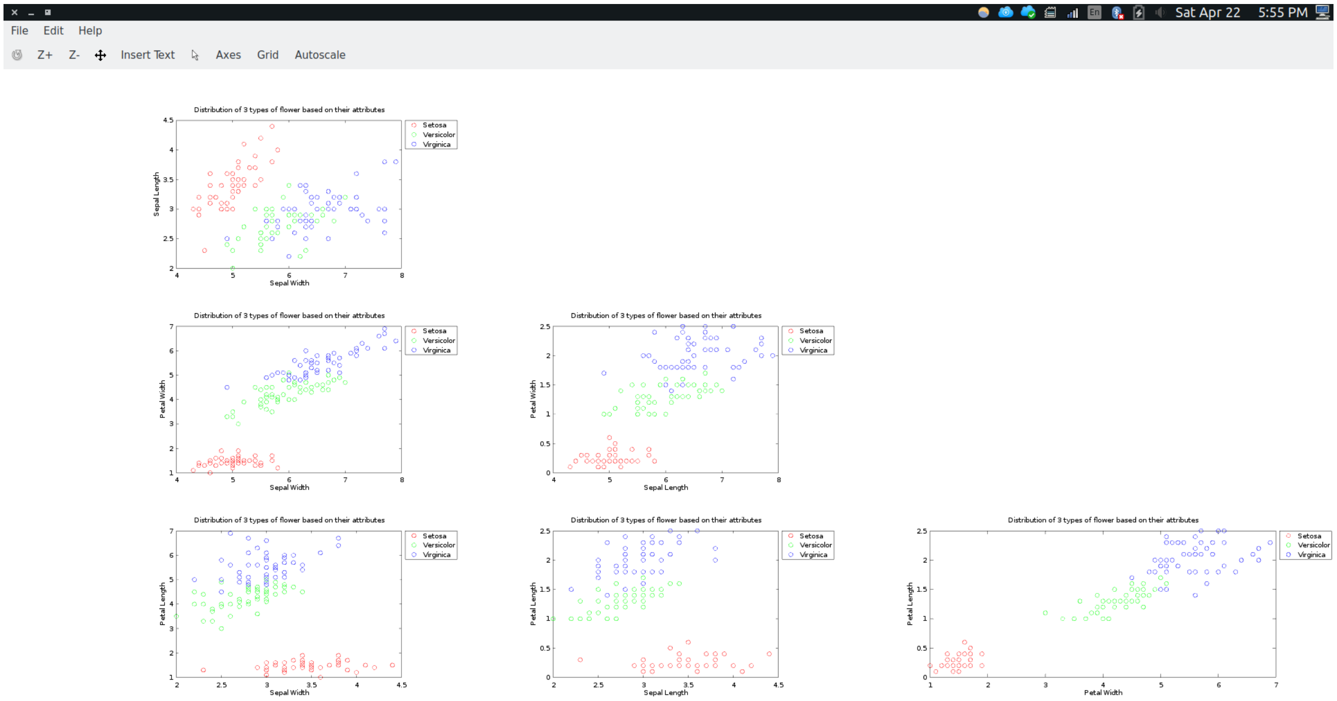


Illustration 2: Plot of the flowers

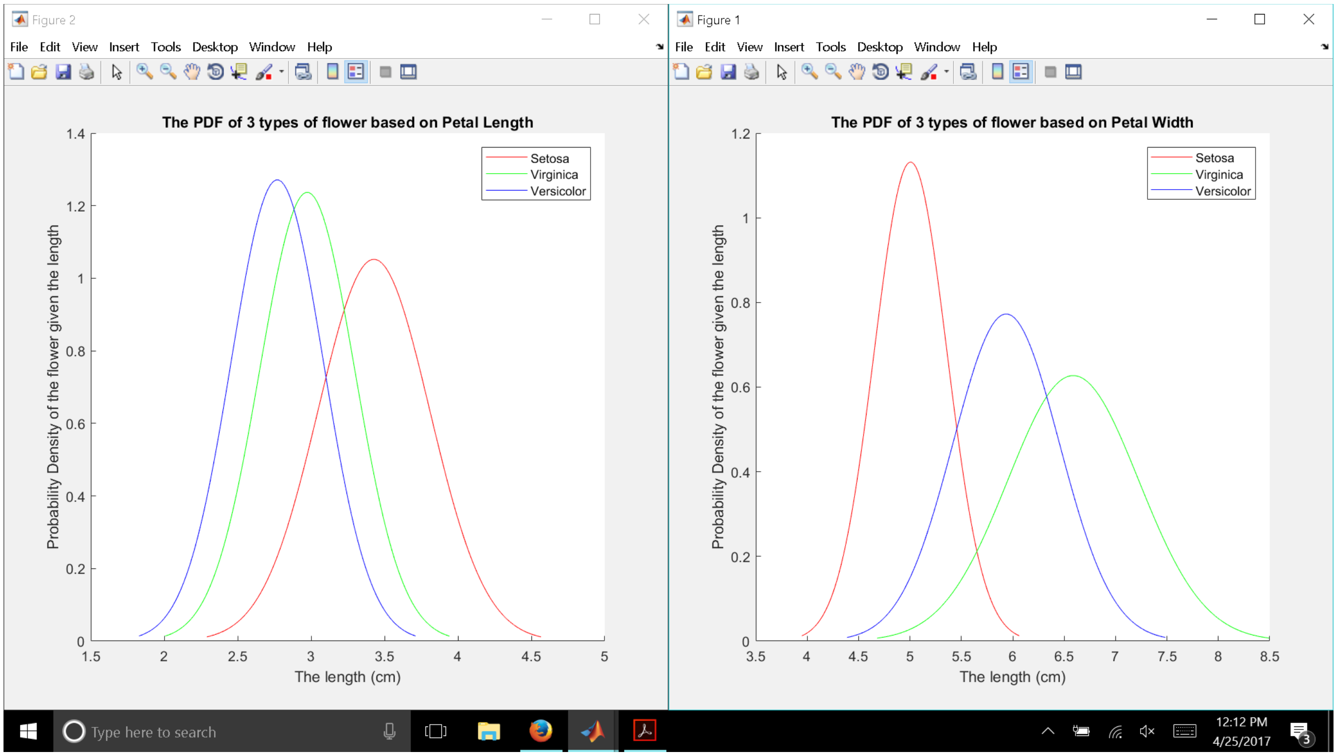


Figure 2
File Edit View Insert Tools Desktop Window Help
The PDF of 3 types of flower based on Petal Length
Probability Density of the flower given the length
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.5
2
2.5
3
3.5
4
4.5
5
The length (cm)
Setosa
Virginica
Versicolor
Figure 1
File Edit View Insert Tools Desktop Window Help
The PDF of 3 types of flower based on Petal Width
Probability Density of the flower given the length
0
0.2
0.4
0.6
0.8
1
1.2
3.5
4
4.5
5
5.5
6
6.5
7
7.5
8
8.5
The length (cm)
Setosa
Virginica
Versicolor
Type here to search
12:12 PM
4/25/2017

Illustration 3: PDF Distribution