CENG 463 Spring 2017

Programming Exercise 1:
Classification using Data with Normal (Gaussian) Distribution

Introduction

In this exercise, you will implement three-class classification using previously labeled data.
It is assumed that the distribution of samples in all classes is normal (Gaussian). First, mean and variances will be extracted from the samples which is called maximum likelihood estimation. Then, classification will be done using the obtained parameters. The Python code to be prepared will be able to classify new samples into Class 0, Class 1 or Class 2.

Files included in this exercise

[**?**] MLE1.py - Python script that will help you through the exercise.
iris.data - Dataset for Exercise 1.
? indicates the files you will need to complete and submit

Throughout the exercise, you will be using the script MLE1.py. This script set up the dataset for the problem and makes calls to functions (if exists any). For this homework, you will need to modify and submit MLE1.py. Please also write your name and student number into MLE1.py.

Submission

You will submit the homework via CMS. There is a homework submission link, clicking to which will lead you a file browser that you can choose your file. Please zip your files (even if there is only one file) before submission and name the file as **yourstudentno-hw1.zip**. Submit the homework until **24 March 2017** 23:55. Late submissions will not be evaluated.

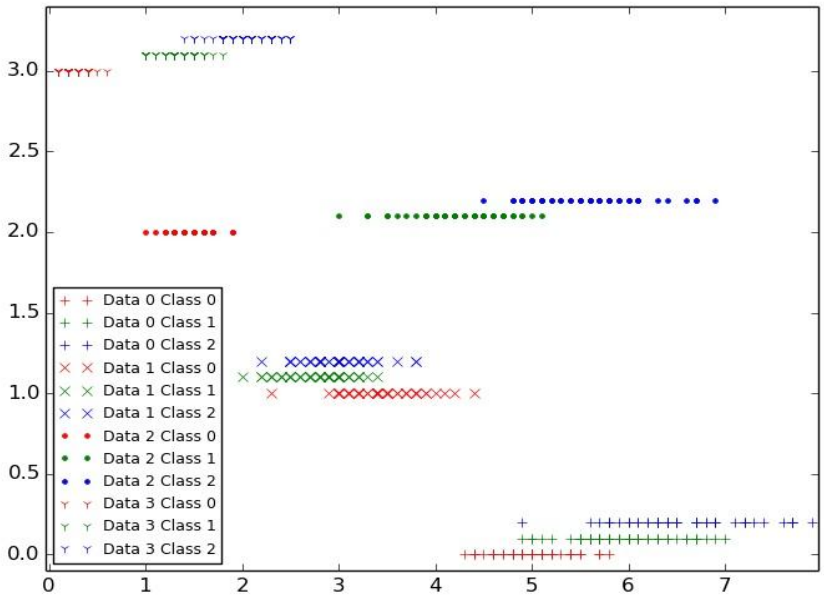
Part 1: Load and Read the Data

The file MLE1.py already contains the script to load and read the data provided. It is the ***Iris* flower data set** or **Fisher's *Iris* data set** which is a multivariate data set introduced by Sir Ronald Fisher in 1936. The data set consists of 50 samples from each of three species of *Iris* (*Iris setosa*, *Iris virginica* and *Iris versicolor*). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

You do not have to write any code for this part of the exercise. Data matrix has five columns; the first four columns contain the samples measurements, whereas the last column contains the labels. The provided code reads the labels.You need to add the code to divide the samples into three vectors, each containing the samples of Class 0, Class 1, and Class 2, respectively.

Part 2: Plotting Data in 1D

This part of the code plots each type of data according to the labels. The code needed for plotting is already in the file. Please examine the code to understand how to assign different colors and signs to the plot. The output should be as follows:



The sample points are not given on a single line for better visualization. Instead, for each class they are slightly shifted.

Part 3: Plotting Data in 2D

This part of the code you should plot one data pair according to the labels in 2D. Examining the plot obtained previously, decide on which pair you would like to use for classification (best candidates) and plot that pair.

Part 4: Extracting Gaussian Parameters

We assume that the given samples suit to a Gaussian distribution. So, we want to compute the maximum likelihood estimates of the parameters (mean, variance and covariance) of these Gaussians. You are expected to find mean, variance and covariance of the samples in each class separately. **Prior probabilites of all classes are assumed to be equal.** Please consult to the lecture notes or go to www.giyf.com if you do not remember the details.
Warning: Throughout this exercise, you can use NUMPY functions mean(), var() and cov() only to validate your own implementation and in case you fail to compute them correctly, to continue the exercise.

Part 5: Plotting Gaussians

Now, we want to see the Gaussian curves corresponding to the parameters we obtained in the previous step. Again, please consult to the lecture notes. For Gaussian function if you need so, go to https://en.wikipedia.org/wiki/Multivariate_normal_distribution. Plot a different surface for each class. Implementation hints are given within the code.

Part 6: Classification

For each sample in the dataset, assign a class label based on your findings. In comments, explain your reasoning behind your implementation with few sentences.

Part 7: Performance Measurement

Calculate and print the percentage of the successfully classified samples.

Part 8: Estimation, Classification and Performance Measurement for Separate Training and Test Samples

Now, we split the dataset into training and test partitions. The code needed for this is already given in the script file. Repeat the Gaussian parameter estimation using the training partition and classification and performance measurement using the test partition. Print the percentage of the successfully classified samples.