EECE5644 Classification Homework #2

Yuxin Lin

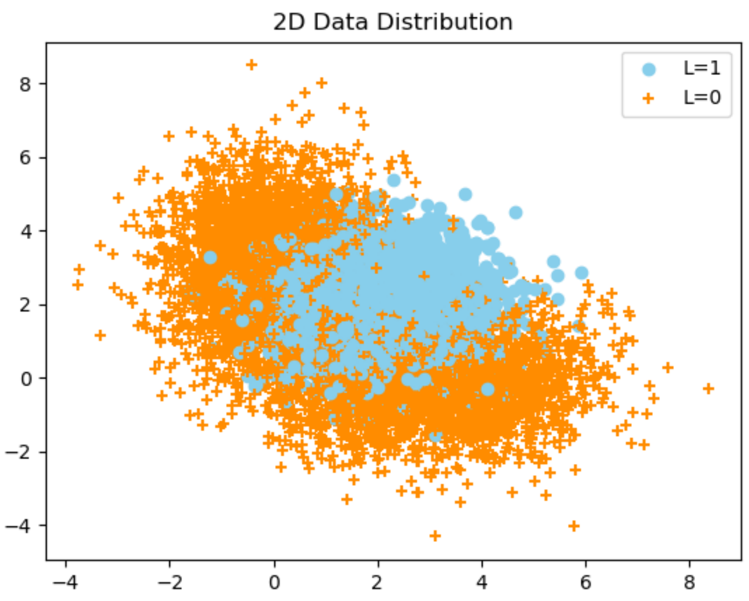
October 07, 2022

Question 1

According to the given information, we can write $p(\mathbf{x})$ as:

$$\begin{aligned} p(\mathbf{x}) &= P(L=0)p(\mathbf{x} \mid L=0) + P(L=1)p(\mathbf{x} \mid L=1) \\ &= 0.65 \times \frac{1}{2} \times (g(\mathbf{x} \mid \mathbf{m}_{01}, \mathbf{C}_{01}) + g(\mathbf{x} \mid \mathbf{m}_{02}, \mathbf{C}_{02})) + 0.35 \times g(\mathbf{x} \mid \mathbf{m}_1, \mathbf{C}_1) \end{aligned}$$

For further calculation, here we simply define $L=1$ as positive, while $L=0$ as negative. The data distribution of the generated samples is as below:



Part A : ERM Classification

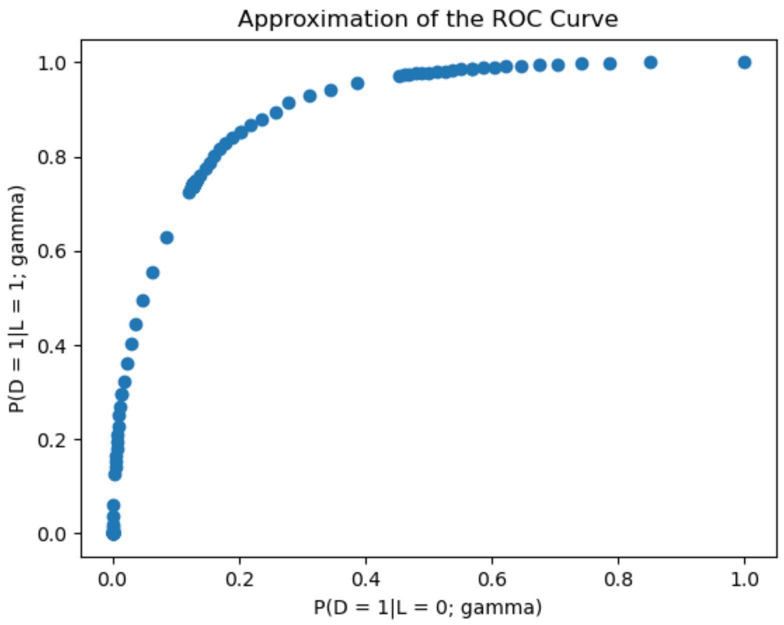
1. The minimum risk solution in the form of a likelihood-ratio test can be written as:

$$\frac{p(\mathbf{x} \mid L=1)}{p(\mathbf{x} \mid L=0)} >^? \gamma = \frac{p(L=0)}{p(L=1)} \frac{\lambda(D=1 \mid L=0) - \lambda(D=0 \mid L=0)}{\lambda(D=0 \mid L=1) - \lambda(D=1 \mid L=1)}$$

   where $\lambda(D=i \mid L=j)$ denotes the loss value for the case $D=i|L=j$.

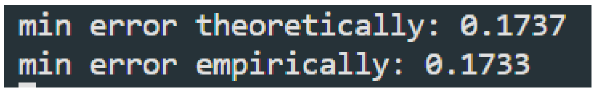
   Therefore, we can say that for a given $\mathbf{x}$, if $\frac{p(\mathbf{x}|L=1)}{p(\mathbf{x}|L=0)} > \frac{0.65}{0.35} \frac{\lambda(D=1|L=0)-\lambda(D=0|L=0)}{\lambda(D=0|L=1)-\lambda(D=1|L=1)}$, we label it as $L=1$, otherwise we label it as $L=0$.

2. Since the threshold value is finite in this problem, here we can produce a γ array with elements gradually growing from 0 to 10000, where 10000 is big enough. After applying an ERM classifier implemented via Python with the γ array on the 10,000 samples we generated before, we can draw an approximation of the ROC curve as below:



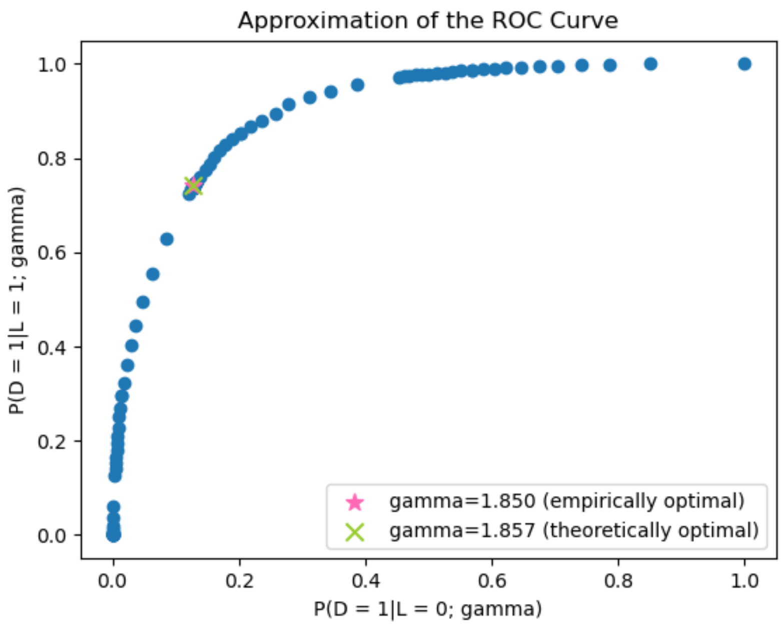
3. Here we consider the situation of 0-1 loss. From 1.A.1, we can derive that the theoretically optimal threshold value to minimize $P(error)$ should be $\gamma = \frac{0.65}{0.35} \times \frac{1-0}{1-0} = 1.857$. Meanwhile, by manually comparing the $P(error)$ calculated directly via different γ on the samples we generated, we can obtain the empirically optimal threshold value: $\gamma = 1.850$. The difference between the two values of γ is very small.

   Apply these two γ on our dataset. We can estimate the minimum probabilities of error both theoretically and empirically. Here we have $P_{min\_theoetically}(error) = 0.1737$ and $P_{min\_empirically}(error) = 0.1733$.



```
min error theoretically: 0.1737
min error empirically: 0.1733
```

   The approximation of the ROC curve with the two operating points is as below:



   From this figure, we can see that the two points representing our two optimal solutions almost overlap each other, which indicates that our theory and practice are compatible. However, according to the calculated values of the $p(error)$, where $0.1733 < 0.1737$, we can find that the empirically optimal threshold works a little bit better than the theoretically optimal threshold.

Part B : LDA Classification

Calculating the class conditional means and covariance matrices on our generated dataset, we have:

$$\mu_0 = \begin{bmatrix} 1.54497502 \\ 1.47352059 \end{bmatrix}, cov_0 = \begin{bmatrix} 3.79872279 & -2.27682895 \\ -2.27682895 & 3.76884968 \end{bmatrix}$$
$$\mu_1 = \begin{bmatrix} 2.00399857 \\ 1.99197958 \end{bmatrix}, cov_1 = \begin{bmatrix} 1.0182181 & 0.00251612 \\ 0.00251612 & 0.97576757 \end{bmatrix}$$

where μ_0, cov_0 denote the mean and the covariance matrix of class 0, and μ_1, cov_1 denote the mean and the covariance matrix of class 1. Comparing to the Gaussian pdfs' parameters for generation, we can say that they are almost the same.

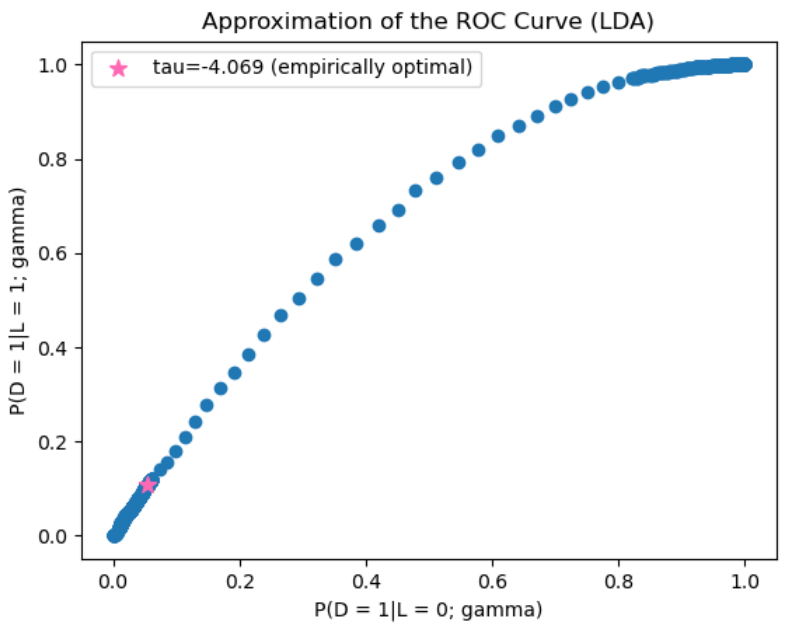
The Fisher LDA projection weight vector $\mathbf{w}_{LDA}$ can be determined via eigen decomposition. In this problem, the eigen vector corresponding to the highest eigen value is $\mathbf{w}^* = \mathbf{w}_{LDA} = \begin{bmatrix} -0.68796778 \\ -0.72574123 \end{bmatrix}$.

To classify the samples using LDA method, firstly we need to project the original samples from 2-dimension to 1-dimension by applying $y = \mathbf{w_{LDA}^T x}$ to every specified $\mathbf{x}$. We go through all the possible threshold τ to determine every sample's label via its y : when $y <= \tau$, decide $L = 1$, otherwise decide $L = 0$ (in this way, the LDA classifier works better than when $y >= \tau$, decide $L = 1$, otherwise decide $L = 0$). In the comparison step, the LDA classifier and ERM classifier are similar in that they both need a certain threshold value for classification. LDA compares the dimensionally reduced data to its threshold, while ERM compares the likelihood ratio to its threshold.

By calculating $p(error)$ at every τ (here we pick finite amount of τ), we can simply determine the best τ empirically that can minimize $p(error)$. In this case, we have $\tau_{optimal} = -4.069$ and $p_{min}(error) = 0.3467$.

```
min error empirically: 0.3467
```

The approximation of the ROC curve with the best τ is as below:



Here we can see that the minimized probability of error of the LDA classifier is much bigger than the one of the ERM classifier we implemented before ($0.1733 < 0.3467$). Meanwhile, the ROC curve generated by the LDA classifier has a smaller AUC than the ERM classifier. Both of the evidences indicate that the performance of the LDA classifier is worse than the ERM classifier.

Question 2

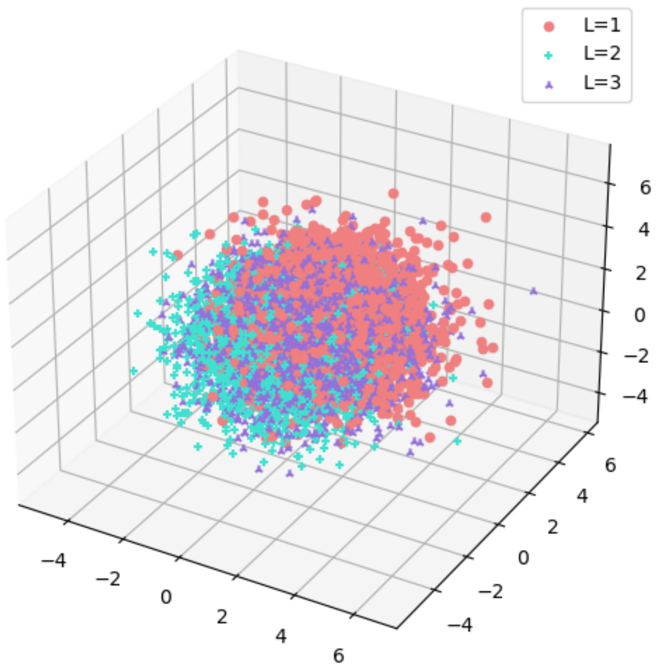
According to the given constrictions, we can simply give 4 Gaussians as below:

$$g_1 = g(\mathbf{x} \mid \begin{bmatrix}1\\1\\1\end{bmatrix}, \begin{bmatrix}2&0&0\\0&2&0\\0&0&2\end{bmatrix}); g_2 = g(\mathbf{x} \mid \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}2&0&0\\0&2&0\\0&0&2\end{bmatrix});$$
$$g_3 = g(\mathbf{x} \mid \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}2&0&0\\0&2&0\\0&0&2\end{bmatrix}); g_4 = g(\mathbf{x} \mid \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}2&0&0\\0&2&0\\0&0&2\end{bmatrix}).$$

where the pdf of Class 1 data can be written as $p(\mathbf{x} \mid L = 1) = g_1$, the pdf of Class 2 data can be written as $p(\mathbf{x} \mid L = 2) = g_2$, and the pdf of Class 3 data can be written as $p(\mathbf{x} \mid L = 3) = \frac{1}{2}(g_3 + g_4)$.

Part A : Minimum Probability of Error Classification

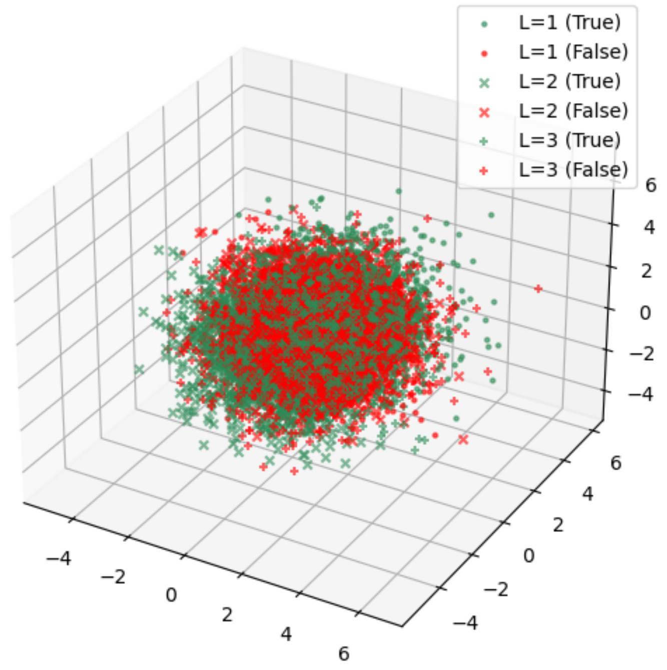
1. The data distribution of the generated samples is as below:



2. Here we consider the situation of 0-1 loss. For a specified $\mathbf{x}$, we can decide its label i by $i = argmax_i \phi_i(\mathbf{x})$, where $\phi_i(\mathbf{x}) = p(\mathbf{x} \mid L = i)P(L = i)$, $i \in \{1, 2, 3\}$. According to this decision rule, we can implement a MAP classifier for multiple classes. Applying the generated dataset on this classifier, we can obtain an empirically estimated confusion matrix as below:

		condition		
		L=1	L=2	L=3
predict	D=1	0.4642	0.0903	0.2311
	D=2	0.1022	0.4670	0.2405
	D=3	0.4336	0.4427	0.5284

3. The data distribution to indicate the classifier's performance is as below:



It's obvious that the central region of the distribution is red, which indicates that samples in the area where their likelihood overlap have a higher probability of being misclassified. At the same time, samples at the edges are more likely to be colored green, which means that they have a lower probability of being misclassified.

Part B : ERM classification

Similarly to Part A, for a specified $\mathbf{x}$, we can decide its label i by $i = argmax_i \phi_i(\mathbf{x})$, where $i \in \{1, 2, 3\}$. Here we have:

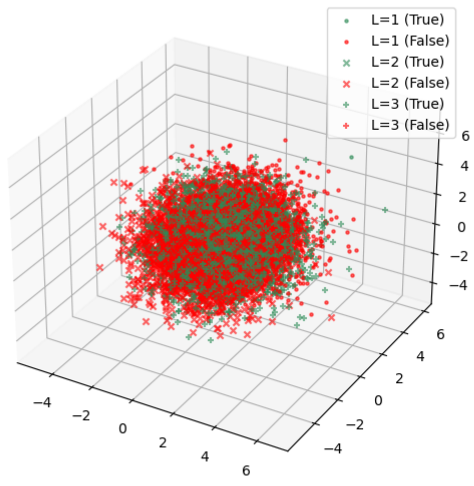
$$\begin{aligned}\phi_i(\mathbf{x}) &= -R(D = i \mid \mathbf{x}) \\ &= -\sum_j \lambda(D = i|L = j) p(L = l \mid \mathbf{x}) \\ &= -\sum_j \lambda(D = i|L = j) \frac{p(\mathbf{x} \mid L = j) P(L = j)}{P(\mathbf{x})}\end{aligned}$$

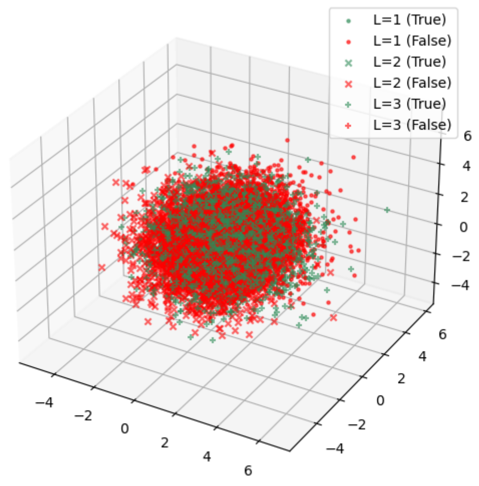
where $\lambda(D = i \mid L = j)$ denotes the $(i, j)^{th}$ entry of the corresponding loss matrix. Since $P(\mathbf{x})$ is the same to every class, here we define $\phi_i'(\mathbf{x})$ as $\phi_i'(\mathbf{x}) = -\sum_j \lambda(D = i|L = j) p(\mathbf{x} \mid L = j) P(L = j)$ for convenience.

To calculate the minimum expected risk, still we need to determine $P(\mathbf{x})$ for each specified $\mathbf{x}$ when making decisions. Here we have $P(\mathbf{x}) = p(\mathbf{x} \mid L = 1)P(L = 1) + p(\mathbf{x} \mid L = 2)P(L = 2) + p(\mathbf{x} \mid L = 3)P(L = 3)$.

Consider the two loss matrices $\Lambda_{10} = \begin{bmatrix} 0 & 1 & 10 \\ 1 & 0 & 10 \\ 1 & 1 & 0 \end{bmatrix}$ and $\Lambda_{100} = \begin{bmatrix} 0 & 1 & 100 \\ 1 & 0 & 100 \\ 1 & 1 & 0 \end{bmatrix}$. This indicates that when a sample's original label is 3, the cost of classifying it wrongly to class 1 or 2 is unbearable. Therefore, our classifier will tend to classify any sample that might belong to class 3 to class 3, since making mistakes on $L = 1$ or 2 is much more acceptable. The things is, the 4 Gaussians we chose for generating our dataset are too close, so that even under the 0-1 loss condition, the probability of misclassifying any sample as class 3 is not small (the confusion matrix in 2.A.2 is a proof). This leads to the fact that no matter using which of the two loss matrices, our classifier will classify almost all the samples to class 3.

The two figures to show the classifier's performance with the two loss matrices are as below:

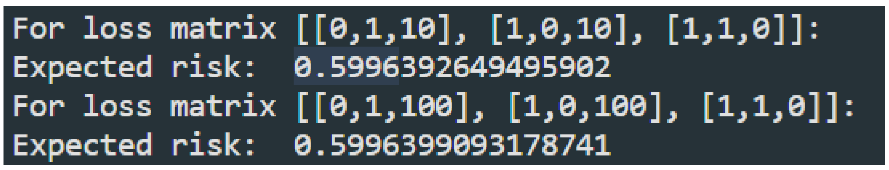




(Left figure: $\Lambda_{10} = \begin{bmatrix} 0 & 1 & 10 \\ 1 & 0 & 10 \\ 1 & 1 & 0 \end{bmatrix}$; Right figure: $\Lambda_{100} = \begin{bmatrix} 0 & 1 & 100 \\ 1 & 0 & 100 \\ 1 & 1 & 0 \end{bmatrix}$)

As mentioned earlier, all the samples belong to class 3 are classified correctly, while most samples belong to class 1 and 2 are classified wrongly. Comparing these two figures to the data distribution in 2.A.1, we can see that the label 3 area is colored green (indicating correct), and the label $1\&2$ area is colored red (indicating incorrect).

In addition, we can calculate that for both of the cases, the estimated minimum expected risks are nearly the same ($R = 0.5996$):



```
For loss matrix [[0,1,10], [1,0,10], [1,1,0]]:
Expected risk:  0.5996392649495902
For loss matrix [[0,1,100], [1,0,100], [1,1,0]]:
Expected risk:  0.5996399093178741
```

Appendix

All the codes can be found in: https://github.com/linyx1998/Intro-MLPR-Homework/tree/main/hw2

In the process of writing the code, I have made references to the formulas in the course slides, the ideas in `generateDataA1Q1.m`, and APIs in https://numpy.org/doc/stable/.

This repository includes a folder "hw2 codes", a folder "hw2 graphs", and a file "hw2 answer" in pdf version. All of the results in "hw2 answer" can be verified by running the code in `main.py` step by step. The purpose of each file or folder in "hw2 codes" is as follows:

- generate sample: sample code and the data generated by it
- 2d_labels.csv: the labels of generated samples for Question1
- 2d_samples.csv: generated samples for Question1
- 3d_labels.csv: the labels of generated samples for Question2
- 3d_samples.csv: generated samples for Question2
- GenerateData.py: generating samples and visualizing the data distribution for Question1 and Question2
- ReadData.py: reading generated data from csv files for Question1 and Question2
- ERMClassifier.py: implementing ERM Classifier for Question1
- LDAClassifier.py: implementing LDA Classifier for Question1

- ROC.py: drawing ROC curves for the ERM Classifier and the LDA Classifier
- MultiClassifier.py: implementing MAP Classifier and ERM Classifier as well as visualizing the data distribution for Question2
- main.py: main function