EECE5644 - Assignment 1

Dhruv Agarwal

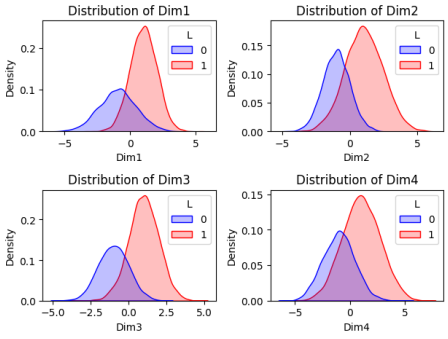
October 2024

1 Question 1

X is a 4-dimensional real-valued random vector with probability distribution function of the form: $p(x) = p(x|L = 0) * p(L = 0) + p(x|L = 1) * p(L = 1)$ where $p(L = 0) = 0.35$ and $p(L = 1) = 0.65$ and $p(x|L = 0)$ and $p(x|L = 1)$ are Gaussian distributions with mean and variance as follows.

$$m_0 = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} c_0 = \begin{bmatrix} 2 & -0.5 & 0.3 & 0 \\ -0.5 & 1 & -0.5 & 0 \\ 0.3 & -0.5 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} m_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} c_1 = \begin{bmatrix} 1 & 0.3 & -0.2 & 0 \\ 0.3 & 2 & 0.3 & 0 \\ -0.2 & 0.3 & 1 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

I generate 10,000 samples of X to be used as data for the classification models, which will be discussed below.



1.1 ERM Classification using the knowledge of the true data pdf

1.1.1 Calculating the threshold for likelihood ratio test as a function of loss and priors

$$\text{Risk}(D = d|x) = \sum_{l=1}^{C} \lambda_{dl} P(L = l|x)$$

where $\lambda_{dl} = \text{Loss}(D = d|L = l)$

$$D(x) = \underset{d\in\{1,\ldots,C\}}{\arg\min}\ R(D = d|x)$$

$$D(x) = \underset{d\in\{0,1\}}{\arg\min}\ \lambda_{d0}P(L = 0|x) + \lambda_{d1}P(L = 1|x)$$

For D=1,

$$\lambda_{00}P(L = 0|x) + \lambda_{01}P(L = 1|x) > \lambda_{10}P(L = 0|x) + \lambda_{11}P(L = 1|x)$$

$$(\lambda_{01} - \lambda_{11})P(L = 1|x) > (\lambda_{10} - \lambda_{00})P(L = 0|x)$$

$$\frac{(\lambda_{01} - \lambda_{11}) * P(x|L = 1) * P(L = 1)}{P(x)} > \frac{(\lambda_{10} - \lambda_{00}) * P(x|L = 0) * P(L = 0)}{P(x)}$$

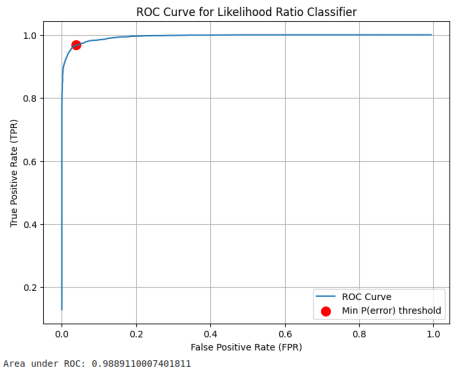
Assuming $\lambda_{01} > \lambda_{11}$

$$\frac{P(x|L = 1)}{P(x|L = 0)} > \frac{(\lambda_{10} - \lambda_{00}) * P(L = 0)}{(\lambda_{01} - \lambda_{11}) * P(L = 1)}$$

$$\text{Threshold} = \frac{(\lambda_{10} - \lambda_{00}) * P(L = 0)}{(\lambda_{01} - \lambda_{11}) * P(L = 1)}$$

1.1.2 Implementing the above classifier

- Implement the likelihood ratio test
- Vary the threshold from 0 to ∞
- Compute the True and False Positives and Negatives
- Plot the ROC curve using the above



Area under ROC: 0.9889110007401811

1.1.3 Analyzing the classifier

Determine the threshold that achieves minimum probability of error

$$P(\text{error};\gamma) = P(D=1|L=0;\gamma)P(L=0) + P(D=0|L=1;\gamma)P(L=1)$$

- Theoretically: $\frac{P(L=0)}{P(L=1)}$
- Empirically: using the ROC curve

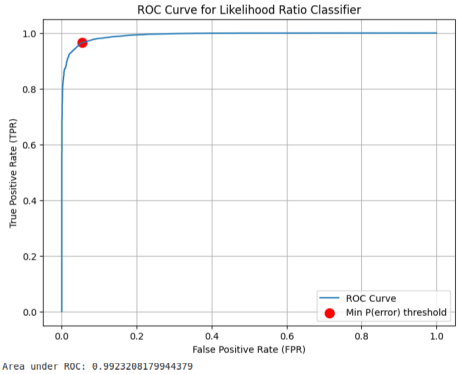
```
Empirical optimal threshold: 0.707701066118189
Minimum probability of error: 0.034374945395046305
Assuming the cost of errors is the same
Theoretical optimal threshold: 0.5384615384615384
```

1.2 ERM Classification using incorrect knowledge of the data distribution

All the steps are the same as in the above part, but we are assuming Naive Bayes assumption that the features are all independent. This means that in the classifier the covariance matrices will change but the data we work with remains the same.

$$c_0 = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \quad c_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

1.2.1 Implementing the above classifier




```
Area under ROC: 0.9923208179944379
```

1.2.2 Analyzing the classifier

```
Empirical optimal threshold: 0.4463233926710397
Minimum probability of error: 0.04100317425148406
Assuming the cost of errors is the same
Theoretical optimal threshold: 0.5384615384615384
```

We can see that the AUROC increases, but the minimum probability of error at the optimal threshold also increases. The increase in AUROC suggests that the model may have become better at differentiating between the two classes, but being better at differentiating the minority class may have led to misclassifying some of the majority points as minority, thus decreasing the accuracy. Furthermore, AUROC compares the models at all thresholds, whereas the accuracy is computed only at the optimal threshold. Model mismatch positively impacts ROC, but negatively impacts the minimum probability of error.

1.3 Fisher LDA Classification

1.3.1 Implementing the above classifier

- Calculating the classwise mean and variance
- Calculate S_W and S_B from the above:

$$S_W = n_0 * cov(L = 0) + n_1 * cov(L = 1)$$

$$S_B = n_0 * (\mu_0 - \mu)(\mu_0 - \mu)^T + n_1 * (\mu_1 - \mu)(\mu_1 - \mu)^T$$

But as it is advised to ignore the difference in n_0 and n_1 we use the equations:

$$S_W = cov(L = 0) + cov(L = 1)$$

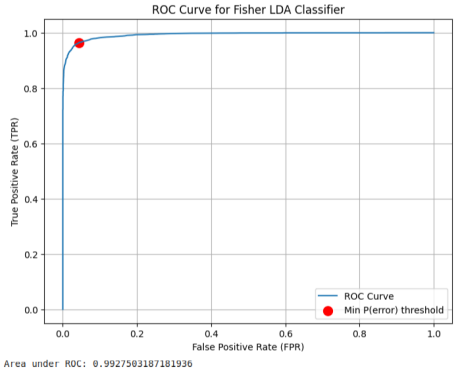
$$S_B = (\mu_0 - \mu)(\mu_0 - \mu)^T + (\mu_1 - \mu)(\mu_1 - \mu)^T$$

- Determining the projection weight vector:

$$J(\theta) = \frac{\theta^T S_B \theta}{\theta^T S_W \theta}$$

$$\max J(\theta) \rightarrow S_B w_{LDA} = \lambda S_W w_{LDA}$$

1. Solve the generalized eigenvector decomposition for $S_W^{-1} S_B$ and compute the eigenvectors and eigenvalues
2. Sort the eigenvalues in descending order
3. Select the eigenvector corresponding to the largest eigenvalue as the optimal projection vector w_{LDA}



1.3.2 Analyzing the classifier

```
Empirical optimal threshold: 0.030795587129142297
Minimum probability of error: 0.03826411987997743
Assuming the cost of errors is the same
Theoretical optimal threshold: 0.5384615384615384
```

We can see that the AUROC increased very slightly and the minimum probability of error also decreased, which suggests that the Fisher LDA is the best model out of the 3. **NOTE:**

- In the first and second part, the threshold was varied from e^{-10} to e^{10}
- In the third part, the threshold was varied from min(projected values) to max(projected values) instead of $-\infty$ to ∞
- $\frac{(\lambda_{10}-\lambda_{00})}{(\lambda_{01}-\lambda_{11})}$ is assumed to be equal to 1

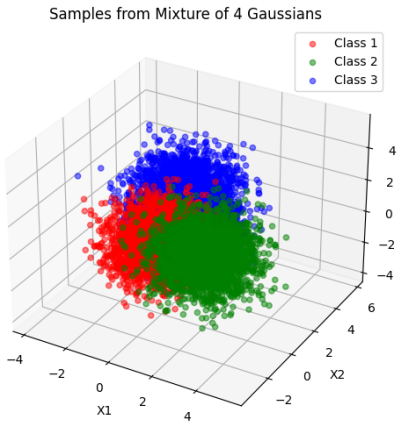
2 Question 2

X is a 3-dimensional real-valued random variable composed of 4 Gaussian distributions. One of them is associated with label 1, another is associated with label 2, and the other 2 are associated with label 3 with probability 0.5 each. We can choose our own Gaussian distributions, but it is advised to keep the distance between means of any two distributions 2 to 3 times of the average standard deviation. I have chosen:

$$m_0 = \begin{bmatrix}0\\0\\0\end{bmatrix} \; m_1 = \begin{bmatrix}2\\0\\0\end{bmatrix} \; m_2 = \begin{bmatrix}0\\2\\0\end{bmatrix} \; m_3 = \begin{bmatrix}0\\2\\2\end{bmatrix}$$

$$c_0 = \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} \; c_1 = \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} \; c_2 = \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} \; c_3 = \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$$

I generate 10,000 samples of X to be used as data for the classification models, which will be discussed below.



2.1 MPE Classification

Decision Rule:

$$f(x|\mu,\Sigma) = \frac{1}{(2\pi)^{k/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$$

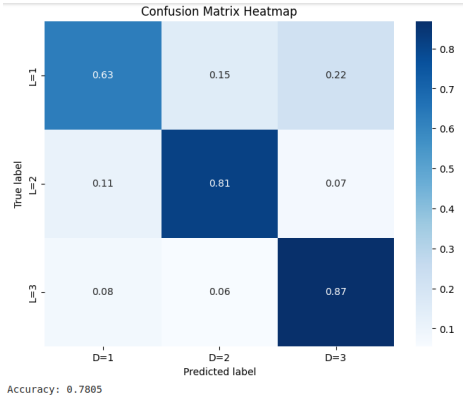
$$\text{post}_{L1} = P_{L1} \cdot f(x|\mu_1,\Sigma_1)$$

$$\text{post}_{L2} = P_{L2} \cdot f(x|\mu_2,\Sigma_2)$$

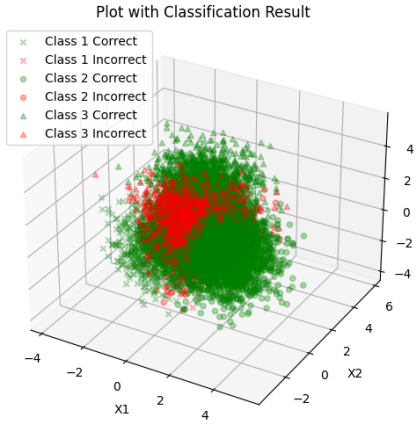
$$\text{post}_{L3} = P_{L3} \cdot (0.5 \cdot f(x|\mu_3,\Sigma_3) + 0.5 \cdot f(x|\mu_1,\Sigma_1))$$

$$\text{Class}(x) = \arg\max(\text{post}_{L1}, \text{post}_{L2}, \text{post}_{L3}) + 1$$

Confusion Matrix:



Plot with classification result:



2.2 ERM Classification

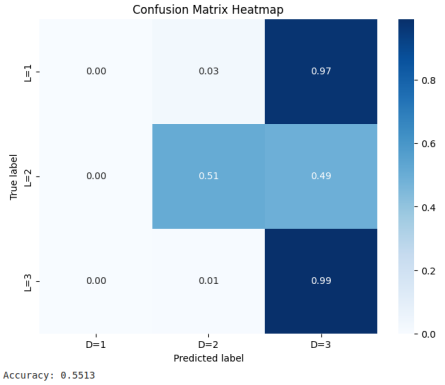
Loss Matrices:

$$\Lambda_{10} = \begin{bmatrix} 0 & 10 & 10 \\ 1 & 0 & 10 \\ 1 & 1 & 0 \end{bmatrix} \Lambda_{100} = \begin{bmatrix} 0 & 100 & 100 \\ 1 & 0 & 100 \\ 1 & 1 & 0 \end{bmatrix}$$

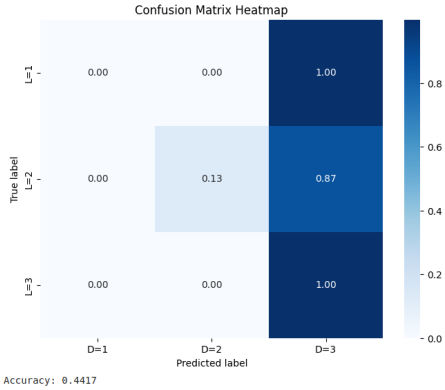
Expected Risk:

```
Expected Risk with Lambda_1: 0.6639916438968279
Expected Risk with Lambda_10: 14.92173737631919
Expected Risk with Lambda_100: 187.05812963666
```

Confusion Matrix for Λ_{10}:



Confusion Matrix for Λ_{100}:



We can see that as the loss for mistaking L=3 is increased, the accuracy decreases and the model starts outputting L=3 for almost all samples.

3 Question 3

Given a test sample X_i and K classes, for each class cls, the discriminant function is defined as:

$$\delta_{\text{cls}}(X_i) = \log\left(\mathcal{N}(X_i \mid \mu_{\text{cls}}, \Sigma_{\text{cls}}) + \epsilon\right) + \log\left(\pi_{\text{cls}} + \epsilon\right)$$

where:

$$\mathcal{N}(X_i \mid \mu_{\text{cls}}, \Sigma_{\text{cls}}) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_{\text{cls}}|}} \exp\left(-\frac{1}{2}(X_i - \mu_{\text{cls}})^T \Sigma_{\text{cls}}^{-1}(X_i - \mu_{\text{cls}})\right)$$

- μ_{cls} is the mean vector for class cls.
- Σ_{cls} is the covariance matrix for class cls.
- π_{cls} is the prior probability for class cls.
- ϵ is a small value to avoid numerical issues.

The classification rule assigns the sample X_i to the class cls_{pred} that maximizes the discriminant function:

$$\text{cls}_{\text{pred}} = \arg\max_{\text{cls}} \delta_{\text{cls}}(X_i)$$

This equation computes the maximum discriminant score for each sample X_i and selects the corresponding class.

Regularization methods used:

$$\Sigma_{\text{cls}} = \frac{1}{n-1}(X_{\text{cls}} - \bar{X}_{\text{cls}})^T (X_{\text{cls}} - \bar{X}_{\text{cls}})$$

Next, compute the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_d$ of the covariance matrix Σ_{cls}, and define the set of non-zero eigenvalues as:

$$\{\lambda_i \mid \lambda_i > 1 \times 10^{-10}\}$$

$$\Sigma_{\text{cls}}^{\text{reg}} = \Sigma_{\text{cls}} + \text{reg\_const} \cdot I_d$$

where I_d is the $d \times d$ identity matrix.

- Geometric Mean Regularization

$$\text{reg\_const} = c_{\text{const}} \cdot \exp\left(\frac{1}{k}\sum_{i=1}^{k} \log(\lambda_i)\right)$$

- Arithmetic Mean Regularization

$$\text{reg\_const} = c_{\text{const}} \cdot \frac{1}{k}\sum_{i=1}^{k} \lambda_i$$

- Constant Regularization

$$\text{reg\_const} = c_{\text{const}}$$

3.1 Dataset 1

3.1.1 Error Probability Estimate

I tested the 3 types of regularization methods with various values of c_{const} and these are the best results I got:

- Constant Regularization; $c_{\text{const}} = 1 * e^{-2}$

```
F1 score (Weighted): 0.5193180137621028
Error Probability Estimate: 0.4654455902724335
```

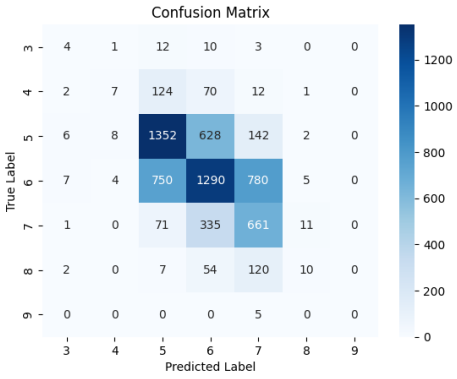
- Geometric Mean Regularization; $c_{\text{const}} = 1 * e^{-1}$

```
F1 score (Weighted): 0.49911643081594165
Error Probability Estimate: 0.48837925196244425
```

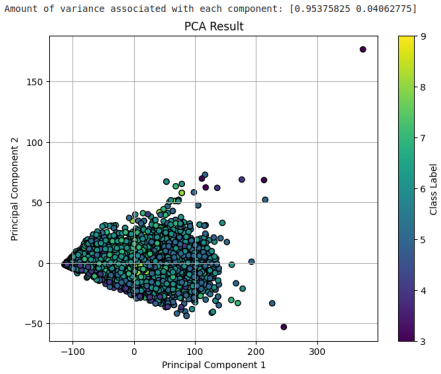
- Arithmetic Mean Regularization; $c_{\text{const}} = 1 * e^{-5}$

```
F1 score (Weighted): 0.5016261448427957
Error Probability Estimate: 0.4914575958134524
```

3.1.2 Confusion Matrix



3.1.3 Principal Component Analysis



3.2 Dataset 2

3.2.1 Error Probability Estimate

I tested the 3 types of regularization methods with various values of c_{const} and these are the best results I got:

- Constant Regularization; $c_{\text{const}} = 1 * e^{-2}$

```
F1 score (Weighted): 0.9413450374823403
Error Probability Estimate: 0.05802511028164237
```

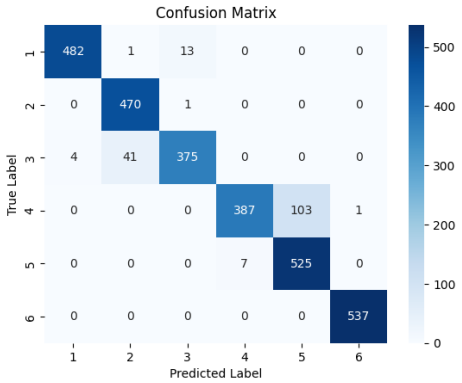
- Geometric Mean Regularization; $c_{\text{const}} = 1 * e^{-2}$

```
F1 score (Weighted): 0.6757346848645563
Error Probability Estimate: 0.28876823888700376
```

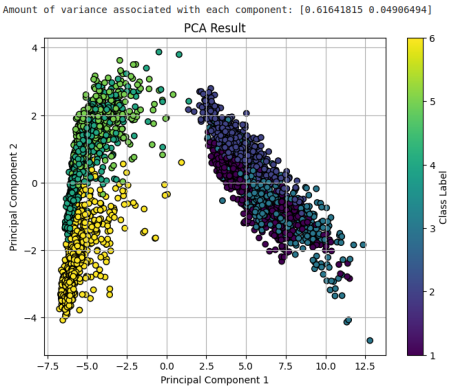
- Arithmetic Mean Regularization; $c_{\text{const}} = 1 * e^{-1}$

```
F1 score (Weighted): 0.7347315530480181
Error Probability Estimate: 0.2273498473023413
```

3.2.2 Confusion Matrix



3.2.3 Principal Component Analysis



- We see that in both cases, Constant regularization gives the best results
- For Dataset 1, all 6497 data points were used instead of just the 4898 mentioned in the question

Link to the Github Folder or copy this into your web browser:
https://github.com/Dhruv-2020EE30592/EECE-5644/tree/main/Assignment-1