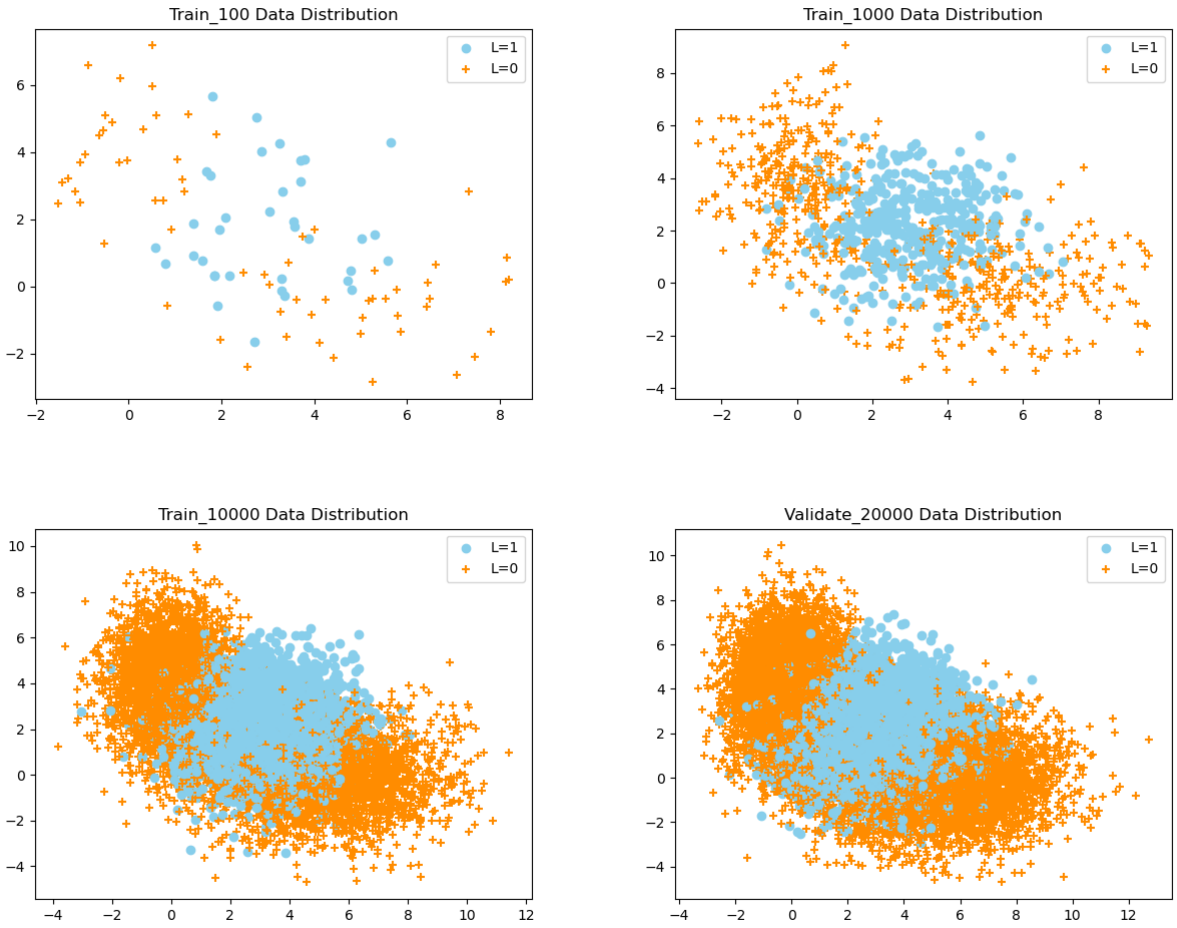
EECE5644 Classification Homework #3

Yuxin Lin

November 11, 2022

Question 1

The data distributions of the generated datasets are shown below:



Part 1

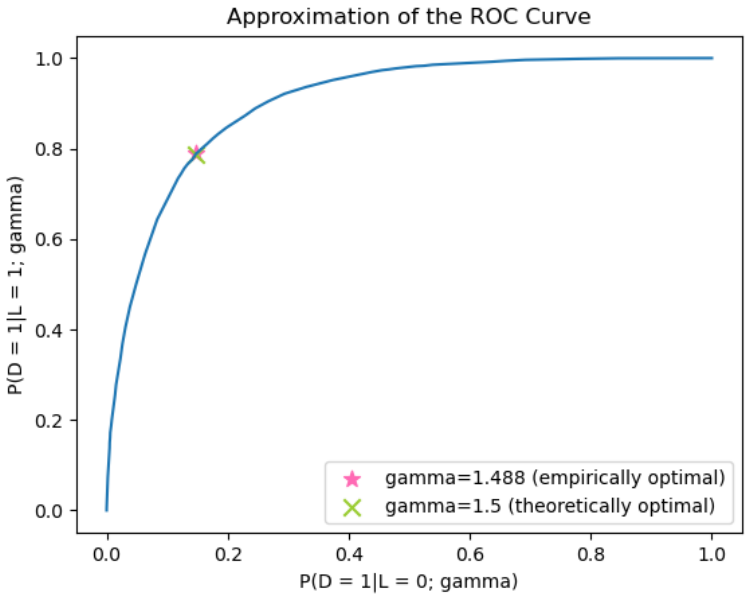
The theoretically optimal classifier can be written as:

$$\frac{p(\mathbf{x} \mid L=1)}{p(\mathbf{x} \mid L=0)} >^{?} \gamma = \frac{p(L=0)}{p(L=1)} \frac{\lambda_{10} - \lambda_{00}}{\lambda_{01} - \lambda_{11}}$$

where λ_{ij} denotes the loss value for the wrong classification case where $D = i | L = j$ (D: Determination).

Here we have $\lambda_{10} = \lambda_{01} = 1$, while $\lambda_{11} = \lambda_{00} = 0$. Therefore, we can say that for a given $\mathbf{x}$, if $\frac{p(\mathbf{x}|L=1)}{p(\mathbf{x}|L=0)} > \frac{0.6}{0.4} = 1.5$, the optimal classifier will label it as $L = 1$, otherwise it will be labeled as $L = 0$.

Implement the classifier and apply it to all the samples in $D_{validate}^{20K}$. The ROC curve of the $min\{P(error)\}$ classifier with special markers indicating the theoretical and the estimated min-error threshold is estimated and plotted as below:

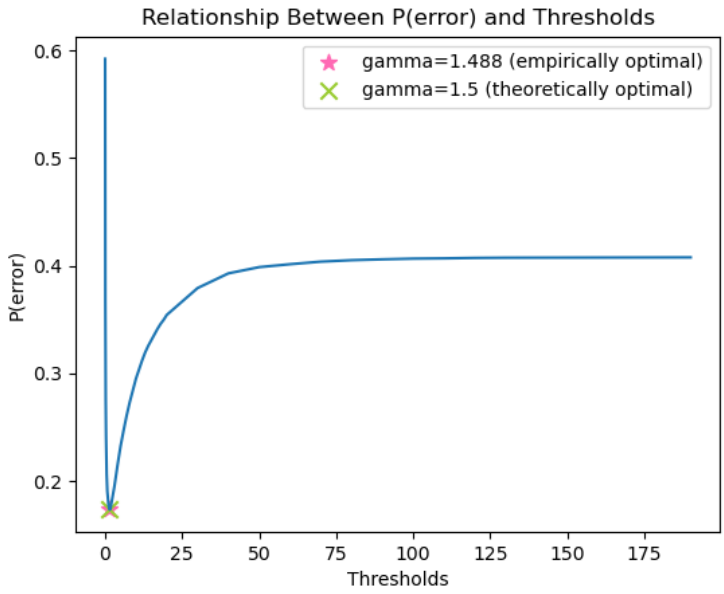


The program running result gives us following information about the achievable minimum probabilities of error corresponding with the two optimal thresholds:

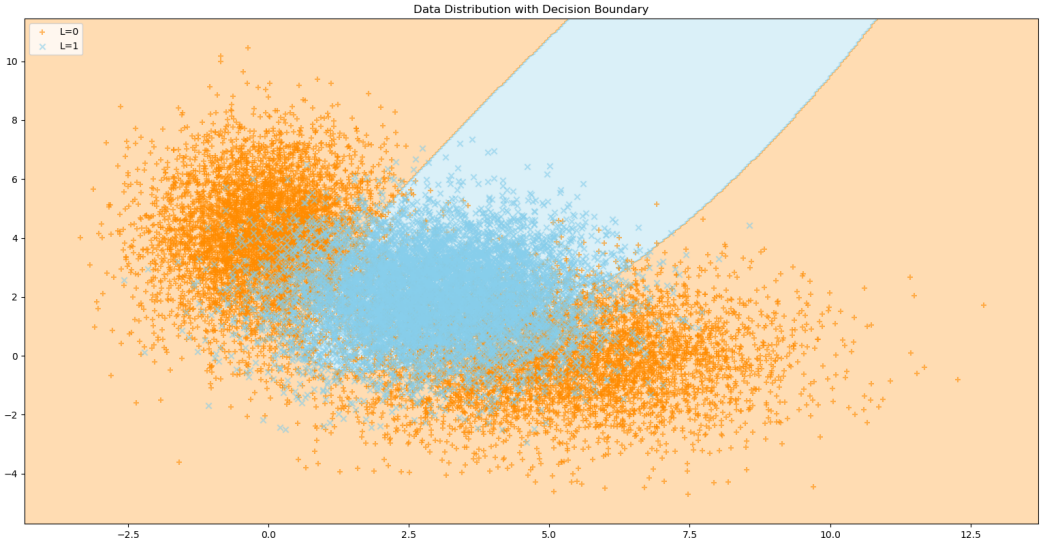
	Threshold	**min{P(error)}**
Theoretically Calculated	1.500	0.1737
Estimated	1.488	0.1735

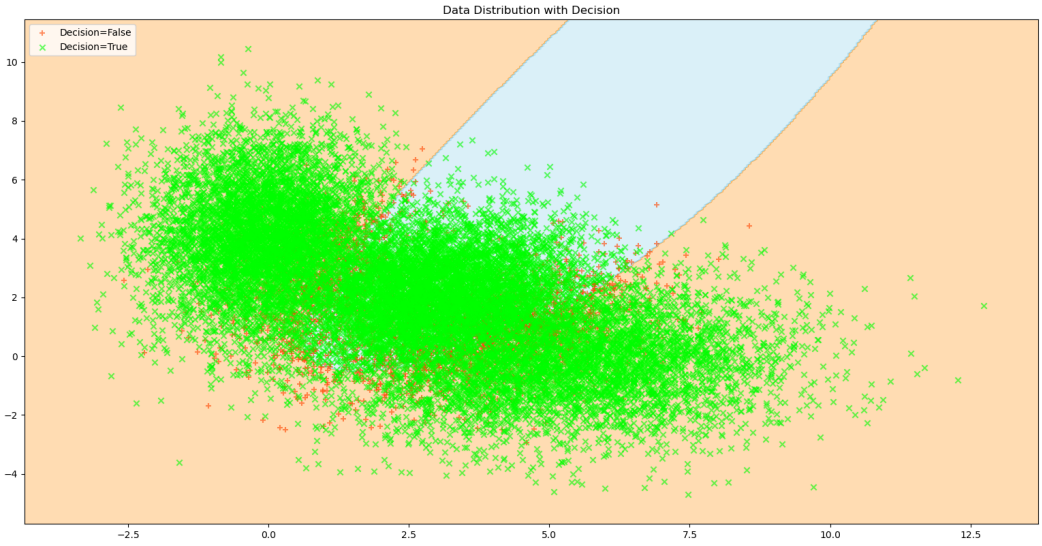
We can see that when the dataset for validation is large enough, the theoretically calculated threshold and the estimated threshold for minimizing the probability of error will be close enough.

To make the result more clear, here we plot another curve to show the relationship between $P(error)$ and the thresholds:



As the supplement, the visualization of the theoretical optimal classifier's decision boundary over the validation dataset is shown below:





Part 2

(a) Calculate the proportion of samples which are labeled as 1 and 0 in D_{train}^{10000} respectively. According to the program running result, the class priors can be estimated as:

$$P(L=1) = 0.402$$
$$P(L=0) = 0.598$$

In this problem, the class conditional pdfs are defined as:

$$p(\mathbf{x} \mid L=1) = g(\mathbf{x} \mid \mathbf{m}_1, \mathbf{C}_1)$$
$$p(\mathbf{x} \mid L=0) = w_1 g(\mathbf{x} \mid \mathbf{m}_{01}, \mathbf{C}_{01}) + w_2 g(\mathbf{x} \mid \mathbf{m}_{02}, \mathbf{C}_{02})$$

where $\mathbf{m}_1, \mathbf{C}_1, \mathbf{m}_{01}, \mathbf{C}_{01}, \mathbf{m}_{02}, \mathbf{C}_{02}$ and ω_1, ω_2 are the parameters that need to be estimated.

Write the two pdfs in a general format $p(\mathbf{x} \mid \boldsymbol{\omega}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\omega}$ denotes the weights of each Gaussian component, $\boldsymbol{\mu}$ denotes the mean vectors of them, and $\boldsymbol{\Sigma}$ denotes the covariance matrices of them. Take the logarithm, then we have:

$$\ln p(\mathbf{x} \mid \boldsymbol{\omega}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \omega_k \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \right\}$$

where N is the size of the dataset, and K is the number of components in the corresponding Gaussian Mixture model.

Use the Expectation-Maximization (EM) algorithm to estimate $\boldsymbol{\omega}$, $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$. We take the E-Step and the M-Step alternatively until convergence:

- **E-Step**

  with current parameters, calculate the probability of each data point $\mathbf{x}_n$ belonging to each component: $\frac{\omega_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \omega_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$. Assign the data points to the component with the highest probability.

- **M-Step**

  According to the assignments in E-Step, update the parameters $\boldsymbol{\omega}$, $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$.

  - Calculate the number of sample points N_k contained in each component k. We have $\omega_k = \frac{N_k}{N}$.
  - $\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N_k} \mathbf{x}_n$, where $\mathbf{x}_n$ denotes data points in component k.
  - $\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N_k} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}} \boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N_k} \mathbf{x}_n$, where $\mathbf{x}_n$ denotes data points in component k.

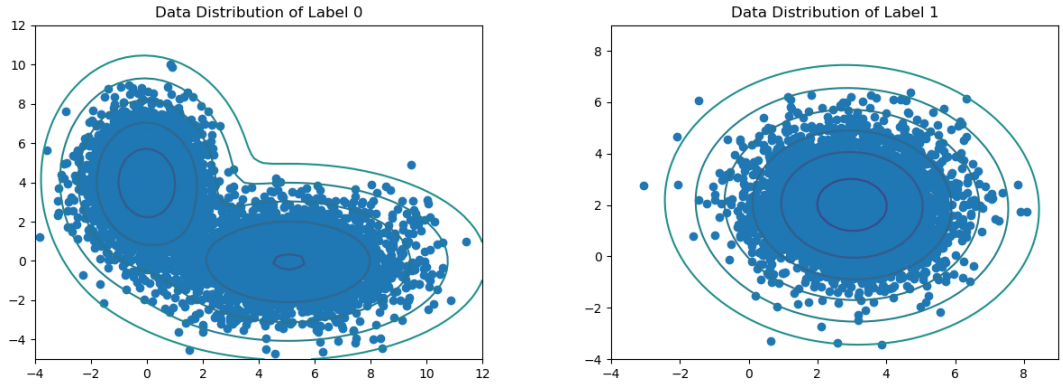
Apply the above algorithm to our dataset D_{train}^{10000} and the assumptions over the data distribution. Then we get:

$$\mathbf{m}_{01} = \begin{bmatrix} 5.085 \\ -0.043 \end{bmatrix} \quad \mathbf{C}_{01} = \begin{bmatrix} 3.850 & 0.017 \\ 0.017 & 1.931 \end{bmatrix} \quad \omega_1 = 0.493$$

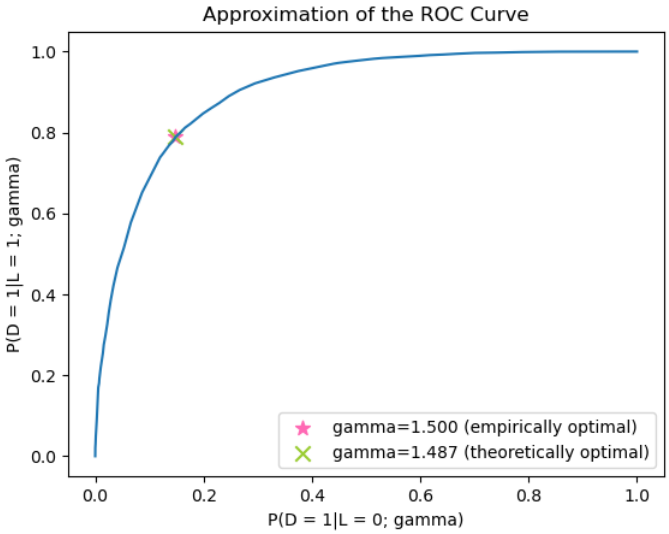
$$\mathbf{m}_{02} = \begin{bmatrix} -0.004 \\ 3.973 \end{bmatrix} \quad \mathbf{C}_{02} = \begin{bmatrix} 1.038 & -0.048 \\ -0.048 & 3.042 \end{bmatrix} \quad \omega_2 = 0.507$$

$$\mathbf{m}_1 = \begin{bmatrix} 3.002 \\ 2.001 \end{bmatrix} \quad \mathbf{C}_1 = \begin{bmatrix} 1.978 & -0.065 \\ -0.065 & 1.979 \end{bmatrix}$$

The estimated distributions of data points with label 0 and label 1 are like:



According to the approximation above, we implement a new $min\{P(error)\}$ classifier. Applying the classifier on dataset $D_{validate}^{20000}$, we can plot the corresponding ROC curve as below:



The achievable minimum probabilities of error corresponding with the theoretical optimal threshold and the estimated optimal threshold are:

	Threshold	**min{P(error)}**
Theoretically Calculated	1.487	0.1736
Estimated	1.500	0.1737

(b) Repeat Part (2a) using D_{train}^{1000} as the training dataset, we get:

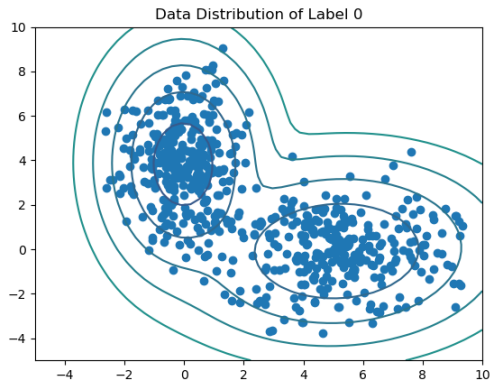
- The class priors:

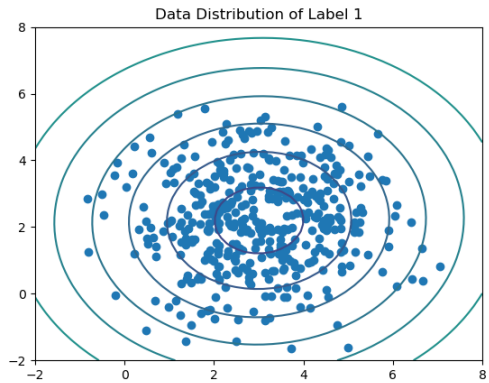
$$P(L=1) = 0.416$$
$$P(L=0) = 0.584$$

- The estimated parameters:

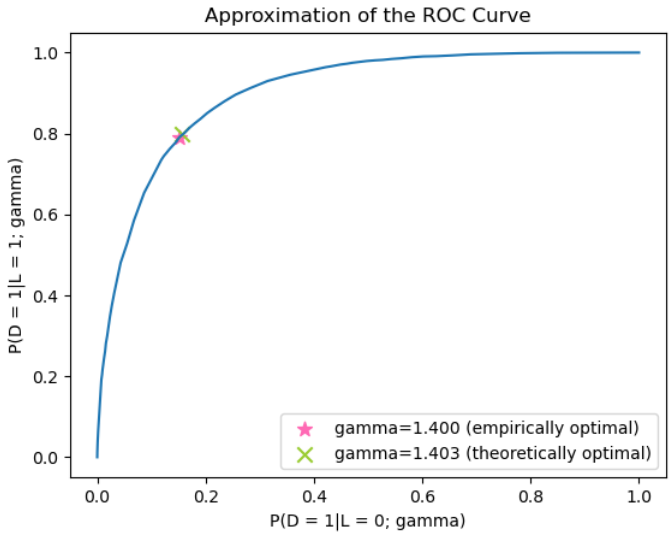
$$\mathbf{m}_{01} = \begin{bmatrix} 5.133 \\ -0.083 \end{bmatrix} \quad \mathbf{C}_{01} = \begin{bmatrix} 3.654 & 0.136 \\ 0.136 & 2.217 \end{bmatrix} \quad \omega_1 = 0.479$$
$$\mathbf{m}_{02} = \begin{bmatrix} -0.035 \\ 3.833 \end{bmatrix} \quad \mathbf{C}_{02} = \begin{bmatrix} 0.985 & -0.017 \\ -0.017 & 3.411 \end{bmatrix} \quad \omega_2 = 0.521$$
$$\mathbf{m}_1 = \begin{bmatrix} 3.008 \\ 2.198 \end{bmatrix} \quad \mathbf{C}_1 = \begin{bmatrix} 1.978 & -0.065 \\ -0.065 & 1.979 \end{bmatrix}$$

- The data distributions:





- The ROC curve over $D_{validation}^{20000}$:



- The achievable minimum probabilities of error:

	Threshold	**min{P(error)}**
Theoretically Calculated	1.403	0.1743
Estimated	1.400	0.1751

(c) Repeat Part (2a) using D_{train}^{100} as the training dataset, we get:

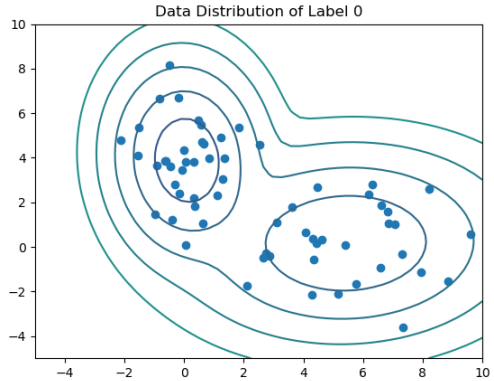
- The class priors:

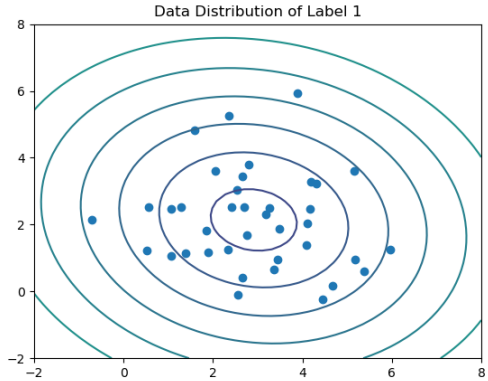
$$P(L=1) = 0.380$$
$$P(L=0) = 0.620$$

- The estimated parameters:

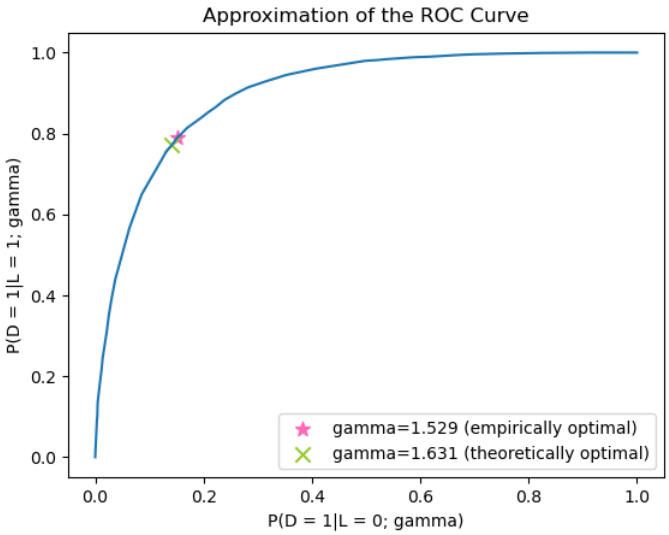
$$\mathbf{m}_{01} = \begin{bmatrix} 5.435 \\ 0.164 \end{bmatrix} \quad \mathbf{C}_{01} = \begin{bmatrix} 4.102 & 0.110 \\ 0.110 & 2.588 \end{bmatrix} \quad \omega_1 = 0.474$$
$$\mathbf{m}_{02} = \begin{bmatrix} 0.085 \\ 3.895 \end{bmatrix} \quad \mathbf{C}_{02} = \begin{bmatrix} 0.970 & -0.095 \\ -0.095 & 2.921 \end{bmatrix} \quad \omega_2 = 0.526$$
$$\mathbf{m}_1 = \begin{bmatrix} 2.912 \\ 2.140 \end{bmatrix} \quad \mathbf{C}_1 = \begin{bmatrix} 1.978 & -0.065 \\ -0.065 & 1.979 \end{bmatrix}$$

- The data distributions:





- The ROC curve over $D_{validation}^{20000}$:



- The achievable minimum probabilities of error:

	Threshold	**min{P(error)}**
Theoretically Calculated	1.631	0.1754
Estimated	1.529	0.1751

According to the experiment results shown above, we can summarize the findings as follows:

1. The larger the size of the dataset used for estimating, the closer the parameters of the data distribution fitted through the EM algorithm will be to the original parameters.
2. The minimum probability of error decreases slightly as the size of the dataset used for estimating increases.

Part 3

In this problem, we need to estimate the approximation of functions based on the logistic regression method. In problem (a), we train a logistic-linear-function-based approximation, while in problem (b), we train a logistic-quadratic-function-based approximation.

According to the *Note 1*, we have:

1. For linear regression, the regression function is $h(\mathbf{x}, \mathbf{w}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{z}(\mathbf{x})}}$, where $\mathbf{z}(\mathbf{x}) = [1, \mathbf{x}^T]^T$,
2. For quadratic regression, the regression function is $h(\mathbf{x}, \mathbf{w}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{z}(\mathbf{x})}}$, where $\mathbf{z}(\mathbf{x}) = [1, x_1, x_2, x_1^2, x_1x_2, x_2^2]^T$,

where $h(\mathbf{x}, \mathbf{w})$ indicates the estimation of $P(\mathbf{Y} = 1|\mathbf{x})$ using the sigmoid function.

Let y_n denotes the determined label of the n^{th} training sample $\mathbf{x}_n$. To choose the optimal parameter $\mathbf{w}$ for regression, we need to specify the optimization problem as minimization of the negative-log-likelihood of the training dataset:

$$\begin{aligned}\mathbf{w} &= \underset{\omega}{argmin}(-\sum_{n=1}^{N}\ln P(y_n|\mathbf{z}(\mathbf{x}_n),\mathbf{w})) \\ &= \underset{\omega}{argmin}(-\sum_{n=1}^{N}(y_n \ln P(y_n=1|\mathbf{z}(\mathbf{x}_n),\mathbf{w}) + (1-y_n)\ln P(y_n=0|\mathbf{z}(\mathbf{x}_n),\mathbf{w}))) \\ &= \underset{\omega}{argmin}(-\sum_{n=1}^{N}(y_n \ln h(\mathbf{x}_n,\mathbf{w}) + (1-y_n)\ln(1-h(\mathbf{x}_n,\mathbf{w}))))\end{aligned}$$

To approach the optimization, we use Python's $minimize$ in the codes. With the estimated class label posteriors, we implement a new $min\{P(error)\}$ classifier by calculate:

$$\gamma = \ln\frac{P(y_n=1|\mathbf{z}(\mathbf{x}),\mathbf{w})}{P(y_n=0|\mathbf{z}(\mathbf{x}),\mathbf{w})} = \mathbf{w}^T\mathbf{z}(\mathbf{x})$$

If $\gamma \geq 0$, we decide $y=1$, otherwise we decide $y=0$.

Calculate the approximation of $P(error)$ over the dataset $D^{20000}_{validate}$, and we get:

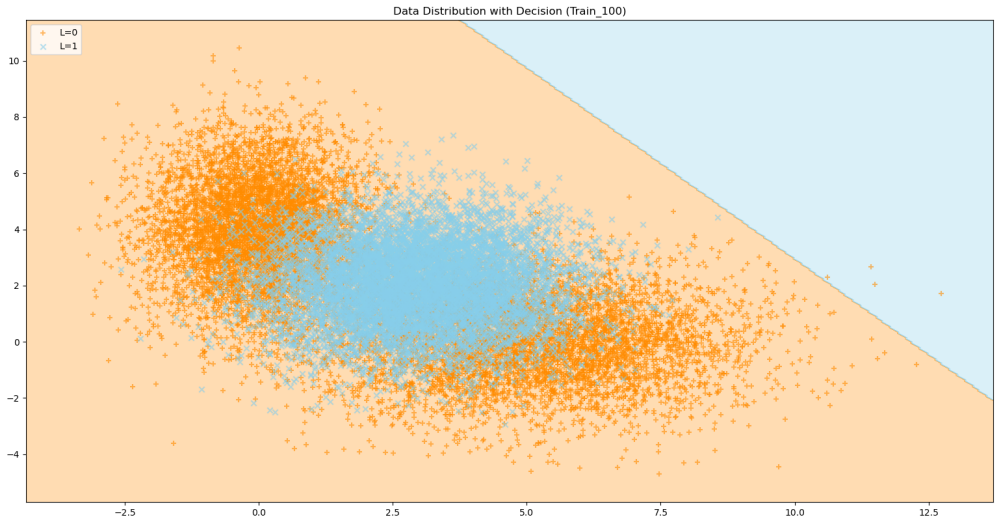
training dataset	Linear Regression	Quadratic Regression
D^{100}_{train}	0.4080	0.2023
D^{1000}_{train}	0.4292	0.2021
D^{10000}_{train}	0.4333	0.1965

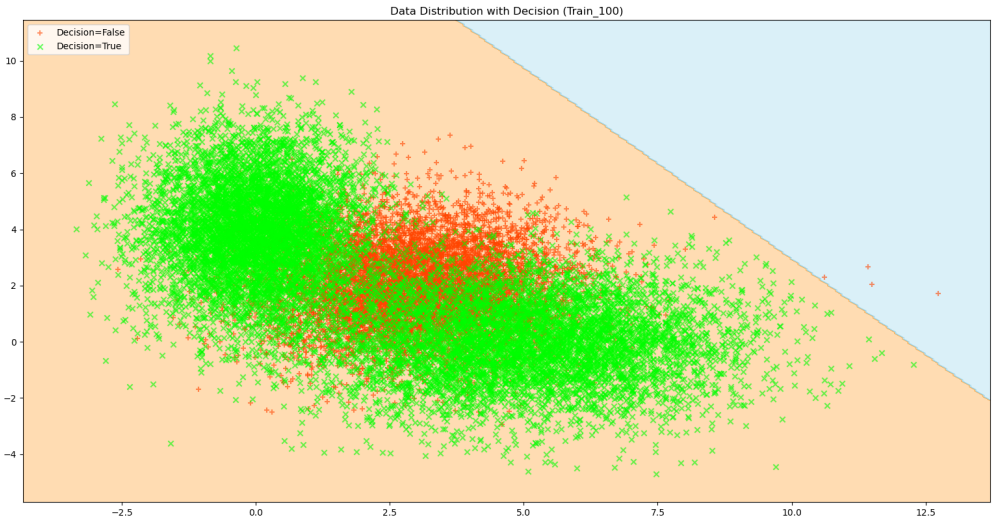
It obvious that linear regression does not match our data distribution at all, which cause the result that the probability of error is high, while the classifier trained with quadratic regression method is much more better than linear regression. For quadratic regression, we can see that the probability of error decreases as the size of training dataset increases. This indicates that more samples can lead to better modeling that can provide better performance.

As the supplement, the visualization of the decision boundaries of the classifiers trained by D^{100}_{train}, D^{1000}_{train} andD^{10000}_{train} over the validation dataset $D^{20000}_{validate}$ is shown below.

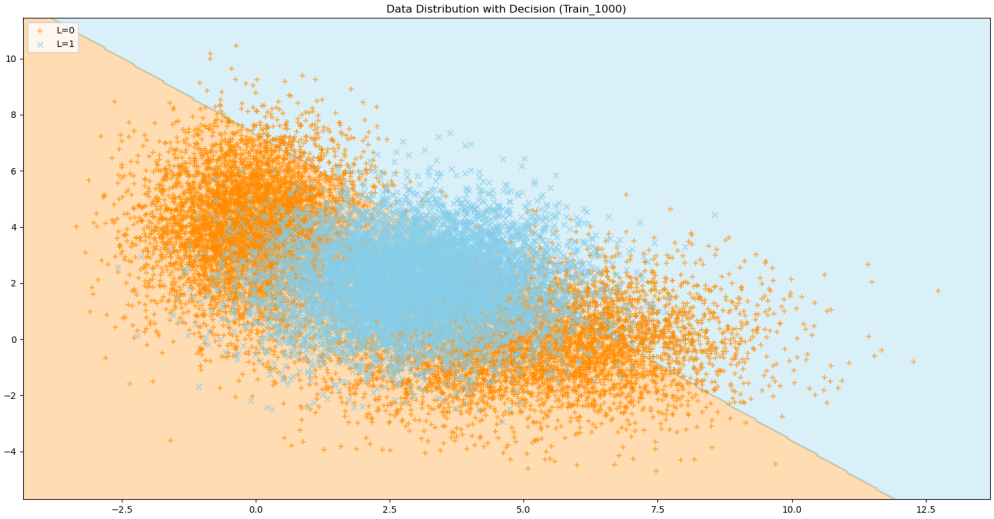
For linear regression, we have:

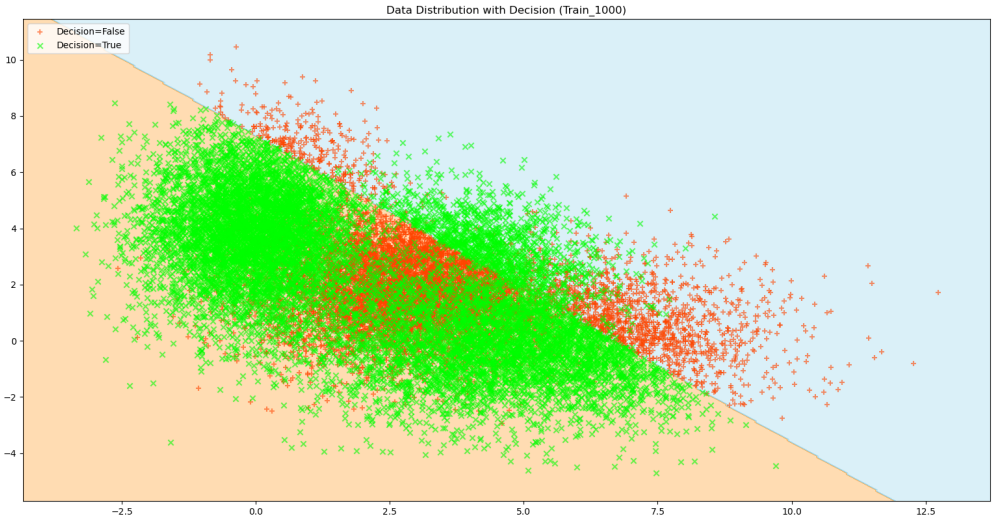
1. D^{100}_{train}



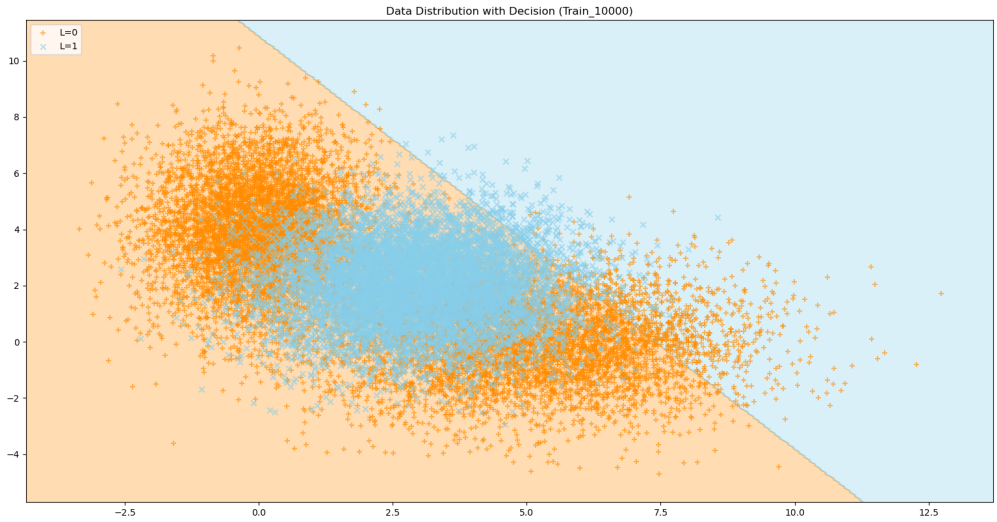


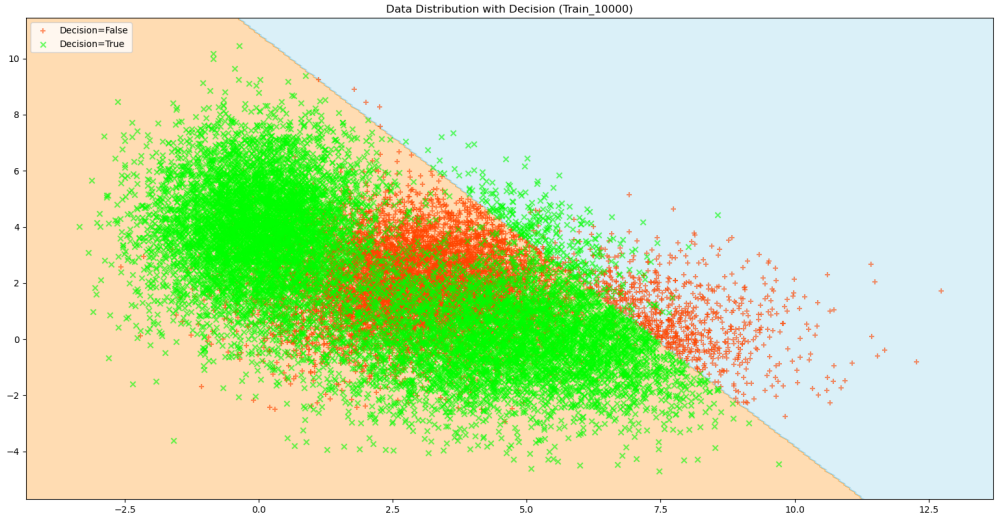
2. D_{train}^{1000}





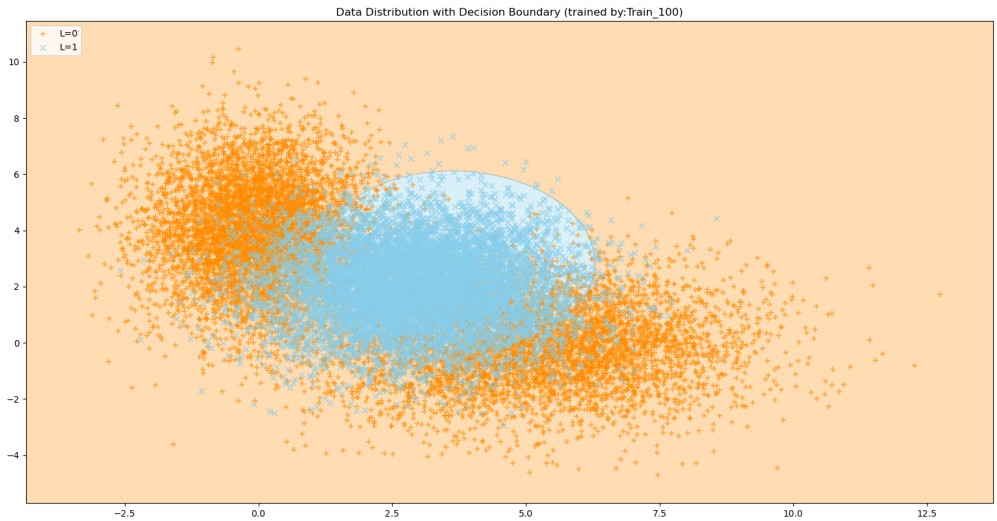
3. D_{train}^{10000}

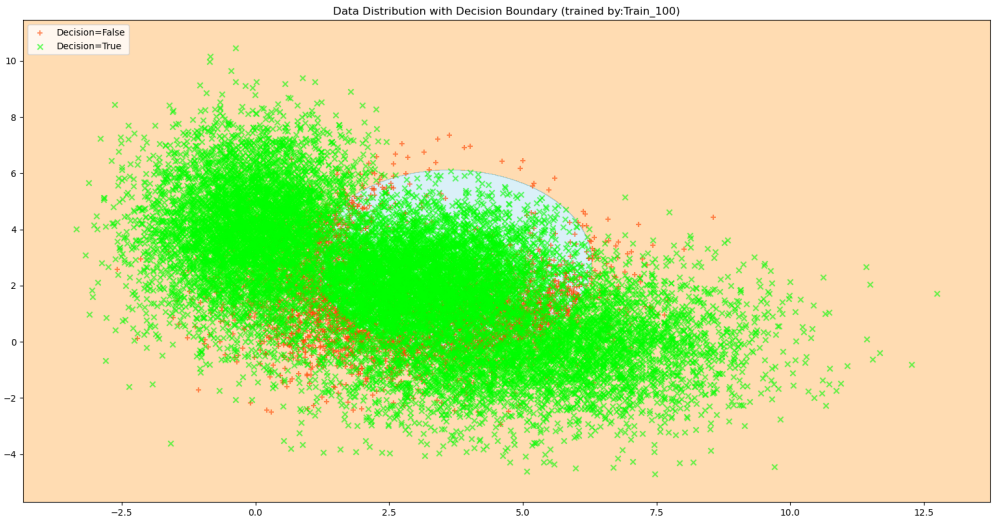




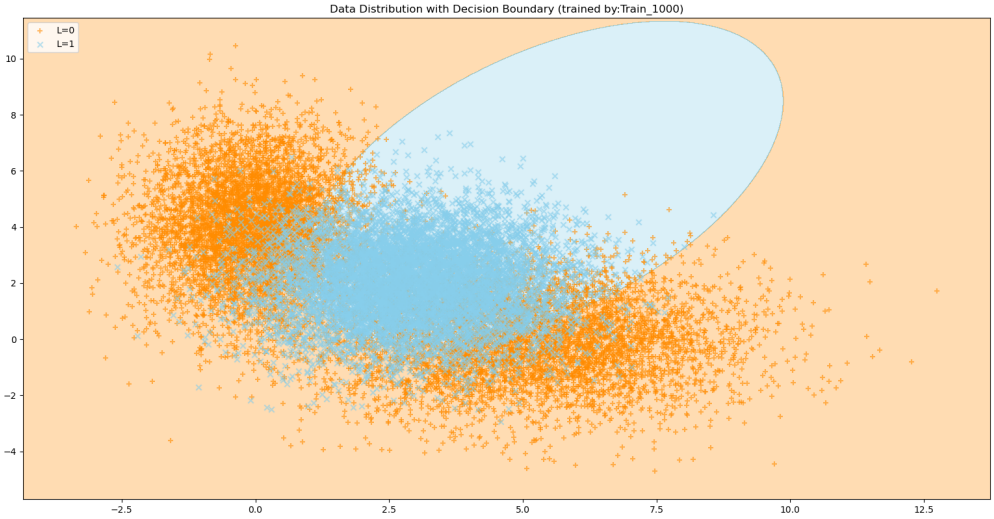
For quadratic regression, we have:

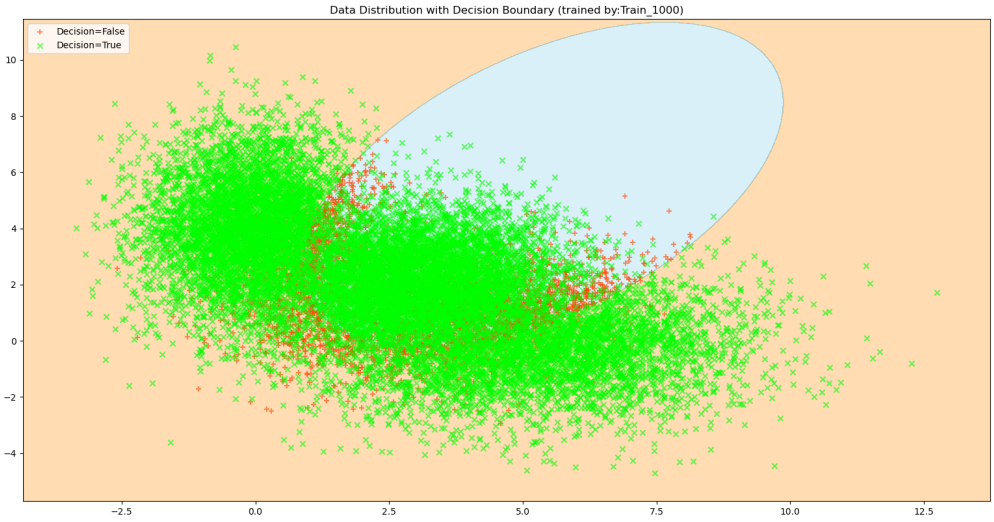
1. D_{train}^{100}



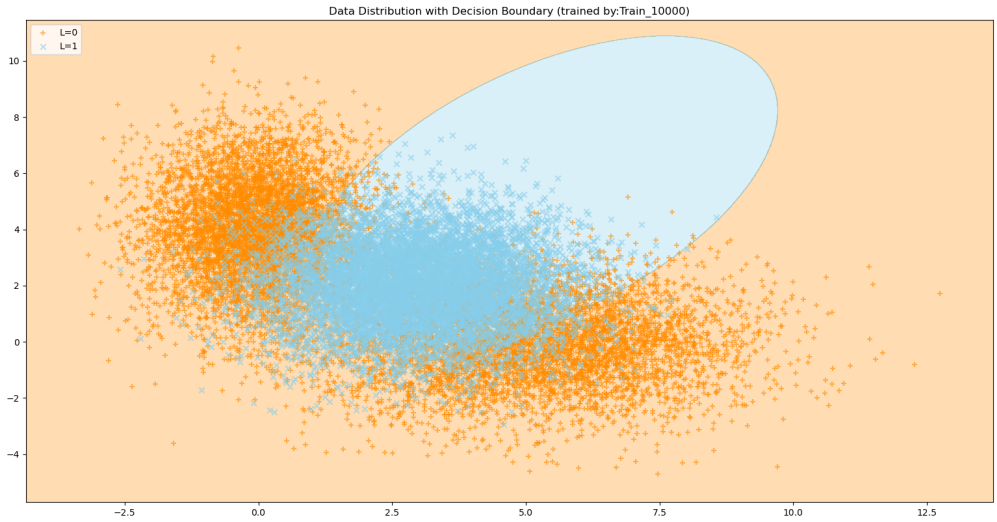


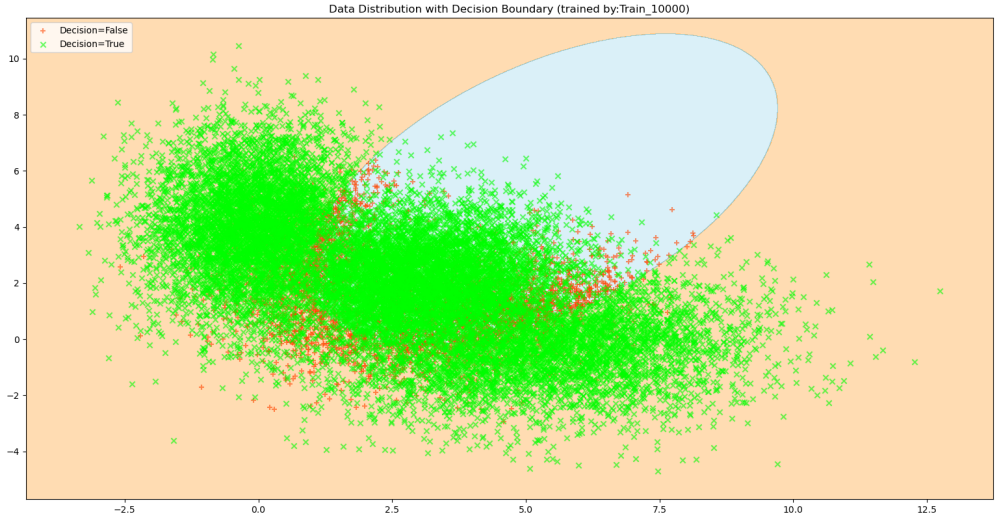
2. D_{train}^{1000}





3. D_{train}^{10000}





Question 2

Express the optimization problem

Given the range measurements $r_1, r_2, \ldots, r_K$, we want to determine the MAP estimate $[x_{MAP}, y_{MAP}]^T$ of the vehicle position. So the optimization problem can be written as:

$$\begin{aligned}
\begin{bmatrix} x_{MAP} \\ y_{MAP} \end{bmatrix} &= \underset{\begin{bmatrix} x \\ y \end{bmatrix}}{argmax} \ln p(\begin{bmatrix} x \\ y \end{bmatrix}, r_1, r_2, \ldots, r_K) \\
&= \underset{\begin{bmatrix} x \\ y \end{bmatrix}}{argmax} \ln p(\begin{bmatrix} x \\ y \end{bmatrix}) p(r_1, r_2, \ldots r_K | \begin{bmatrix} x \\ y \end{bmatrix}) \\
&= \underset{\begin{bmatrix} x \\ y \end{bmatrix}}{argmax} (\ln p(\begin{bmatrix} x \\ y \end{bmatrix}) + \sum_{i=1}^{K} \ln p(r_i | \begin{bmatrix} x \\ y \end{bmatrix}))
\end{aligned}$$

According to the assumption of the prior knowledge of the vehicle position, we have:

$$\ln p(\begin{bmatrix} x \\ y \end{bmatrix}) = (2\pi\sigma_x\sigma_y)^{-1} e^{-\frac{1}{2}\begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix}}$$

$$= -\ln(2\pi\sigma_x\sigma_y) - \frac{1}{2}\begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix}$$

where $-\ln(2\pi\sigma_x\sigma_y)$ is a constant, which does not contribute to $argmax$. In the following process, all the constant term will be removed.

According to the definition of r_i, we have:

$$\sum_{i=1}^{K} \ln p(r_i|\begin{bmatrix} x \\ y \end{bmatrix})) = \sum_{i=1}^{K} \ln p(d_i + n_i|\begin{bmatrix} x \\ y \end{bmatrix}))$$

Since n_i is a zero mean Gaussian measurement, we can say that the distance $d_i = n_i - r_i$ between $[x, y]^T$ and $[x_i, y_i]^T$ is Gaussian distributed with mean r_i and known variance σ_i^2.

Therefore, we can simplify the optimization problem as below:

$$\begin{bmatrix} x_{MAP} \\ y_{MAP} \end{bmatrix} = \underset{\begin{bmatrix} x \\ y \end{bmatrix}}{argmax}(-\frac{1}{2}\begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix} + \sum_{i=1}^{K} \ln \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(d_i - r_i)^2}{2\sigma_i^2}})$$

$$= \underset{\begin{bmatrix} x \\ y \end{bmatrix}}{argmax}(-\frac{1}{2}\begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix} - \sum_{i=1}^{K} \frac{(d_i - r_i)^2}{2\sigma_i^2})$$

$$= \underset{\begin{bmatrix} x \\ y \end{bmatrix}}{argmax}(-\begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} 1/\sigma_x^2 & 0 \\ 0 & 1/\sigma_y^2 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} - \sum_{i=1}^{K} \frac{(d_i - r_i)^2}{\sigma_i^2})$$

$$= \underset{\begin{bmatrix} x \\ y \end{bmatrix}}{argmin}(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} + \sum_{i=1}^{K} \frac{(d_i - r_i)^2}{2\sigma_i^2})$$

To sum up, the simplified optimization problem that needs to be solved is:

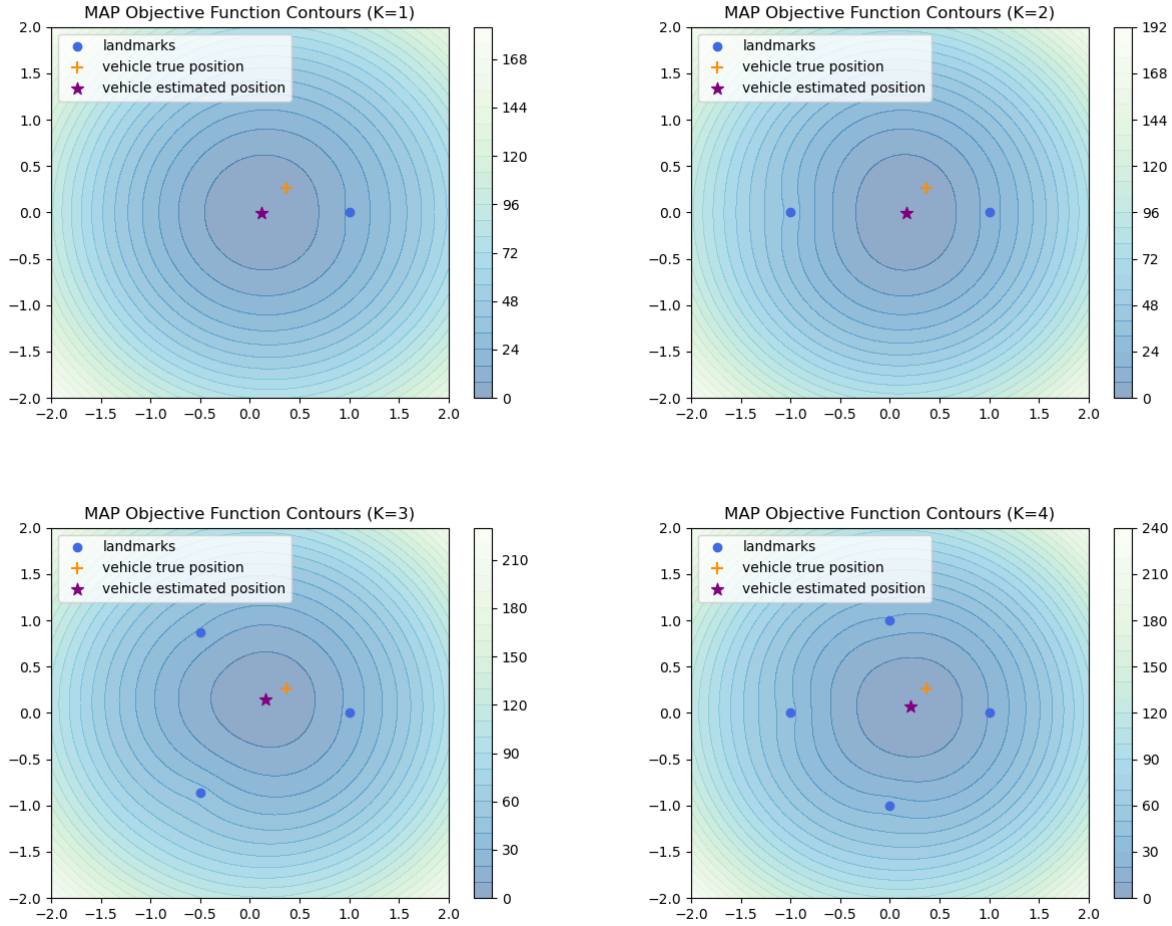
$$\begin{bmatrix} x_{MAP} \\ y_{MAP} \end{bmatrix} = \underset{\begin{bmatrix} x \\ y \end{bmatrix}}{argmin}(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} + \sum_{i=1}^{K} \frac{(d_i - r_i)^2}{2\sigma_i^2})$$

Implementation with computer code

Set $\sigma_x = 0.25$, $\sigma_y = 0.25$, and the noise standard deviation $\sigma_1 = \sigma_2 = \ldots = \sigma_K = 0.3$.

For implementing, we first generate K landmarks on a circle with unit radius centered at the origin, and then the true position of the vehicle inside the circle. After that, randomly generate K range measurements according to the Gaussian model stated above for estimation.

Based on our expression of the optimization problem, we can plot the MAP objective function contours for each K from 1 to 4 as following:



where the landmarks are marked with 'o', the true position of the vehicle is marked with '+', and the estimation of the vehicle position with MAP method is marked with '*'. The estimation is obtained by minimizing the MAP objective function.

Here we can see that the estimated position of the vehicle is more likely to appear inside the circle used for initialization, while outside this circle, the probability of the vehicle's appearance drops more and more sharply (the contour lines have become more dense there).

To make the relationship between the vehicle position and the estimated position more clear, calculate the distance between them, and then we get:

K=1	K=2	K=3	K=4
0.3694	0.3363	0.2457	0.2566

We can speculate from all the running results that as the number of landmarks increases, the distance between the vehicle position and the estimated position tend to decrease, although $d_{K=4} < d_{K=3}$ in this instance. That might due to the fact that the maximum value of K is so small that the experiment is not general enough. Still, we believe that the MAP estimate will get closer to the true position as K increases, since more samples in the training dataset will lead to a more accurate model.

Question 3

First of all, consider classifying a data point $\mathbf{x}$ to one of the c classes. Assume that $P(\omega_i|\mathbf{x})$ is not smaller than any $P(\omega_j|\mathbf{x}), j = 1, \ldots, c$, the risk of classifying $\mathbf{x}$ can be written as:

$$R_{\mathbf{x}} = \begin{cases} \lambda_r, & \text{if reject} \\ \lambda_s(1 - P(\omega_i|\mathbf{x})), & \text{if classify to } \omega_i \\ \lambda_s(1 - P(\omega_j|\mathbf{x})), & \text{if classify to } \omega_j, \text{ where } i \neq j \end{cases}$$

To make the best determination, we take the optimal action whose risk is the lowest, so we need to find $min\{\lambda_r, \lambda_s(1 - P(\omega_i|\mathbf{x})), \lambda_s(1 - P(\omega_j|\mathbf{x}))\}$ for any $j \neq i$.

Since $P(\omega_i|\mathbf{x})$ is not smaller than any $P(\omega_i|\mathbf{x}), i = 1, \ldots, c$, we have $\lambda_s(1 - P(\omega_i|\mathbf{x})) \leq \lambda_s(1 - P(\omega_j|\mathbf{x}))$ for any $j \neq i$, which means that if $\mathbf{x}$ is not rejected, we decide ω_i if $P(\omega_i|\mathbf{x}) \geq P(\omega_j|\mathbf{x})$ for all j.

Then, if $\lambda_r < \lambda_s(1 - P(\omega_i|\mathbf{x}))$, which is $P(\omega_i|\mathbf{x}) < 1 - \frac{\lambda_r}{\lambda_s}$, we reject it as being unrecognizable anyway, otherwise if $P(\omega_i|\mathbf{x}) \geq 1 - \frac{\lambda_r}{\lambda_s}$, we still decide ω_i as stated above.

Therefore, the statement that the minimum risk is obtained when we decide ω_i if $P(\omega_i|\mathbf{x}) \geq P(\omega_j|\mathbf{x})$ for all j and if $P(\omega_i|\mathbf{x}) \geq 1 - \frac{\lambda_r}{\lambda_s}$, and reject otherwise has been proved.

If $\lambda_r = 0$, we have:

1. If $P(\omega_i|\mathbf{x}) = 1$, both rejection and determination on ω_i are acceptable. In this situation, rejection costs nothing, while the classifier is 100% sure that $\mathbf{x}$ should be classified to class i.
2. If $P(\omega_i|\mathbf{x}) < 1$, $P(\omega_i|\mathbf{x}) < 1 - \frac{\lambda_r}{\lambda_s} = 1$ is always satisfied. According to our discussion above, we reject it anyway.

If $\lambda_r > \lambda_s$, we have $1 - \frac{\lambda_r}{\lambda_s} < 0$. Therefore $P(\omega_i|\mathbf{x}) < 1 - \frac{\lambda_r}{\lambda_s} < 0$ will never be satisfied: we will never reject any $\mathbf{x}$, and we will classify it into a class only based on the posterior probability. Here we decide ω_i if $P(\omega_i|\mathbf{x}) \geq P(\omega_j|\mathbf{x})$ for all j.

Appendix

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