Anush Sriram Ramesh

Machine Learning and Pattern Recognition EECE 5644 V30

HW 3

MLE, MAP and Cost minimization

GitHub Link **– Go into folder HW3 for code and results**
Data generated is stored in "Train_10000Samples.csv", "Train_1000Samples.csv" and "Train_100Samples.csv".

Question 1.

1.PART (1) Theoretically Optimal Classifier for Minimum P of Error

From HW2, the theoretically optimal classifier for the dataset is likelihood ratio test.

$$\frac{P(X|L=1)}{P(X|L=0)} \gtrless \frac{P(L=0)}{P(L=1)} \times \frac{\lambda_{10}-\lambda_{11}}{\lambda_{01}-\lambda_{00}} \quad \textit{Equation 1}$$

Where, λ is the cost matrix associated with the decisions.
Let us assume [0 – 1] cost matrix for minimizing the probability of error.

	ACTUAL TRUTH		
DECISIONS	**LABEL**	0	1
	0	$\lambda_{00}=0$	$\lambda_{01}=1$
	1	$\lambda_{10}=1$	$\lambda_{11}=0$

Substituting values of λ back into the equation 1,

$$\frac{P(X|L=1)}{P(X|L=0)} \gtrless \frac{P(L=0)}{P(L=1)}$$

Data extracted from the question,
$P(X|L=0) = w_1 g(m_{01}, c_{01}) + w_2 g(m_{02}, c_{02})$
$P(X|L=1) = g(m_1, c_1)$,
where, $w_1 = w_2 = 0.5$ and means, covariances parameters as per the question.

$$\frac{P(X|L=1)}{P(X|L=0)} \gtrless \frac{0.6}{0.4}$$

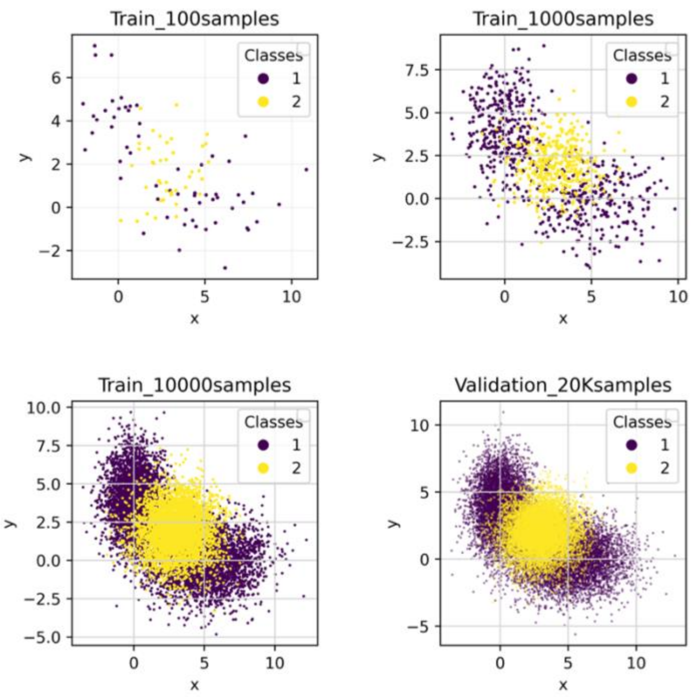


Fig: Training samples generated as per the likelihood function. Class 1 is P(X|L=0) and Class 2 is P(X|L=1)

ROC Curve for Theoretically optimal classifier:

From the above equation, Probability of error and threshold corresponding to the probability can be calculated. It can also be found empirically by varying the threshold in the code.

Table 1

	THEORETICAL	EMPERICAL
PROBABILITY OF ERROR	0.175	0.16328
THRESHOLD	1.5	1.41

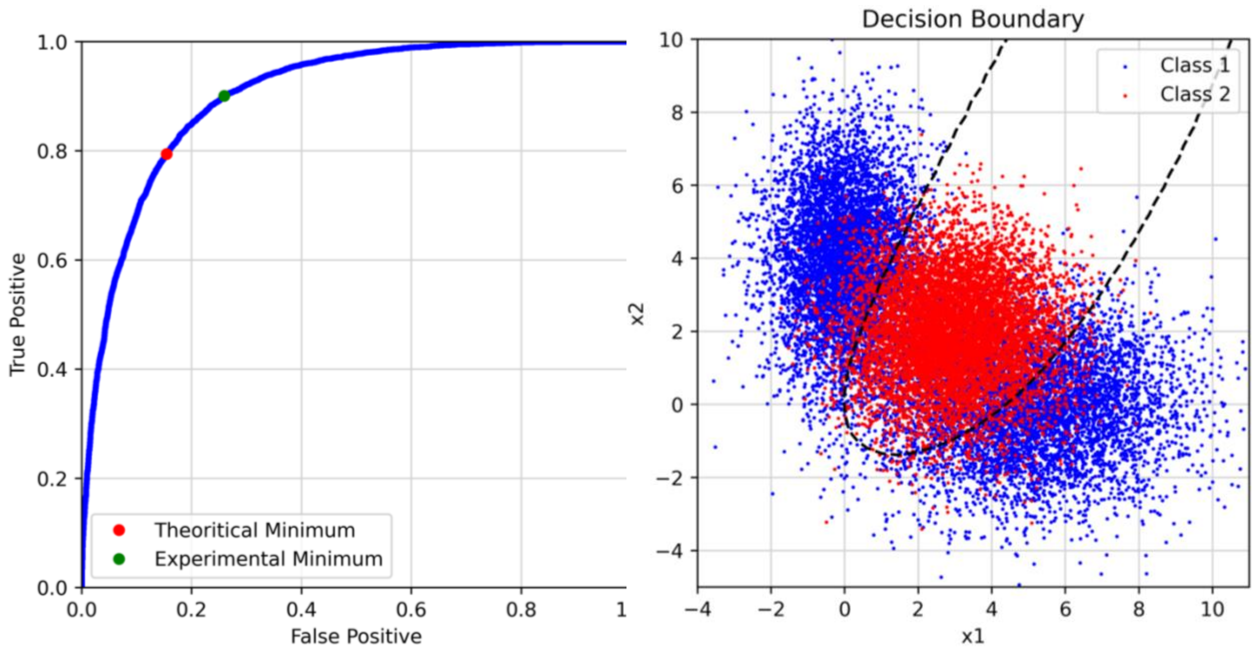


Fig: ROC Curve generated for likelihood ratio classifier with theoretical and empirical thresholds for minimum P of error marked. OPTIONAL SOLUTION Best classifier decision boundary achieved by the likelihood ratio test.

The AUC for the curve is high (close to 1) indicating that the classifier performs well on the data.

OPTIONAL: It is apparent from the contour of the decision boundary that it indeed is a very good classifier for the given data.
We've to try to implement an estimated classifier which performs close to this classifier.

1.PART (2) Parameter estimation for Gaussian Mixture Model using EM Algorithm

We ignore the parameters mentioned in the question for this part. We will try to estimate the class priors, mean and covariance for class labels 0 and 1.

Estimating Parameters for Class 0

Estimating parameters for class 0 involved estimating the weights of the gaussian (w_1, w_2) and the corresponding means and covariances (m_{01}, c_{01}) and (m_{02}, c_{02}).

EM algorithm estimates the parameters of the GMM by maximizing the likelihood function given the prior estimation Θ_g (parameters of GMM) .

$$argmax\ \theta \sum_{i=1}^{M}\sum_{i=1}^{N} lnw_i p\left(l\middle|x_i,\theta_g\right) + \sum_{i=1}^{M}\sum_{i=1}^{N} ln\ w_i\ p\left(l\middle|x_i,\theta_g\right)\ ln\ p_l\ (x_i|\theta_l)$$

Here, θ_g is the prior estimate of the model parameters. θ vector contains ($w_1, w_2, \dots w_m, m_{01}, c_{01}, m_{02}, c_{02} \dots m_m, c_m$) – Formal definition of EM estimator.

Updates used by the estimator are

For Class 0,

$$w_l = \frac{1}{N}\sum_{i=1}^{N} ln\ w_i\ p\left(l\middle|x_i,\theta_g\right)$$

$$\mu_l = \frac{\sum_{i=1}^{N} x_i ln\ w_i\ p\left(l\middle|x_i,\theta_g\right)}{\sum_{i=1}^{N} ln\ w_i\ p\left(l\middle|x_i,\theta_g\right)}$$

$$\Sigma_l = \frac{\sum_{i=1}^{N} ln\ w_i\ p\left(l\middle|x_i,\theta_g\right)(x_i-\mu_l)(x_i-\mu_l)^T}{\sum_{i=1}^{N} ln\ w_i\ p\left(l\middle|x_i,\theta_g\right)}$$

For Class 1, there is only one gaussian. Hence the mean and covariance are the sample mean and covariance.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N} (x_i - \mu)(x_i-\mu)^T$$

Class priors are estimated simply as the percentage of samples in Class label 0 and label 1 as we know the labels for samples in training data.

$$\text{Prior 1} = \frac{Number\ of\ Samples\ in\ Class\ label\ 0}{Total\ Number\ of\ Samples}\quad and$$

$$Prior\ 2 = \frac{Number\ of\ Samples\ in\ Class\ label\ 1}{Total\ Number\ of\ Samples}$$

Results for Training MLE with 10K Samples

Using the **GMM** in **SKLearn.mixture** in python to fit a gaussian mixture model to the samples. Below figures show the contours plotted with the estimated gaussian parameters received from that function.

Class Label 0

$\mathrm{M}_{01} = [5.05078929 \quad -0.03789508].$

$\mathrm{M}_{02} = [0.0385675 \qquad 4.00982853]$ and $\mathrm{w}_1 = [0.50223738 \qquad 0.49776262]$

$$\mathrm{C}_{01} = \begin{pmatrix} 3.9764 & 0.1100 \\ 0.1100 & 1.9484 \end{pmatrix}\quad and\ \mathrm{C}_{02} = \begin{pmatrix} 1.0012 & -0.0119 \\ -0.0119 & 2.9532 \end{pmatrix}\ and\ Prior\ 1 = 0.597$$

Class Label 1
$\mathrm{M}_2 = [3.01594145 1.99267388]$

$$\mathrm{C}_2 = \begin{pmatrix} 1.9998 & 0.0189 \\ 0.0189 & 2.0498 \end{pmatrix} \; and\ Prior\ 2 \; = \; 0.403$$

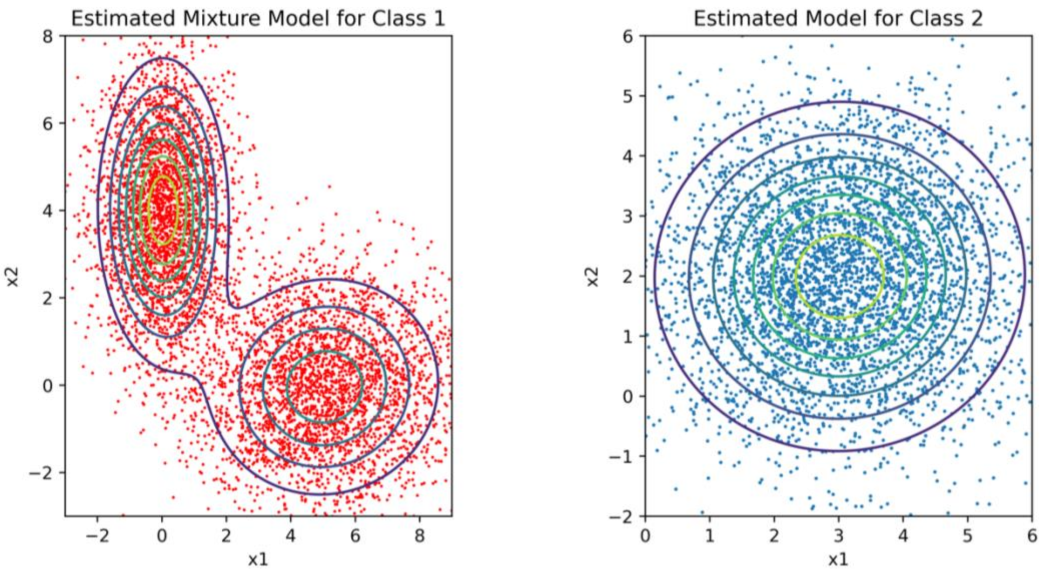


Fig: Contour plots of the estimated gaussian distributions for each class label generated by EM algorithm.

Using the estimated gaussian parameters run the Likelihood ratio classifier on the data and report the ROC Curve and Probability of error.

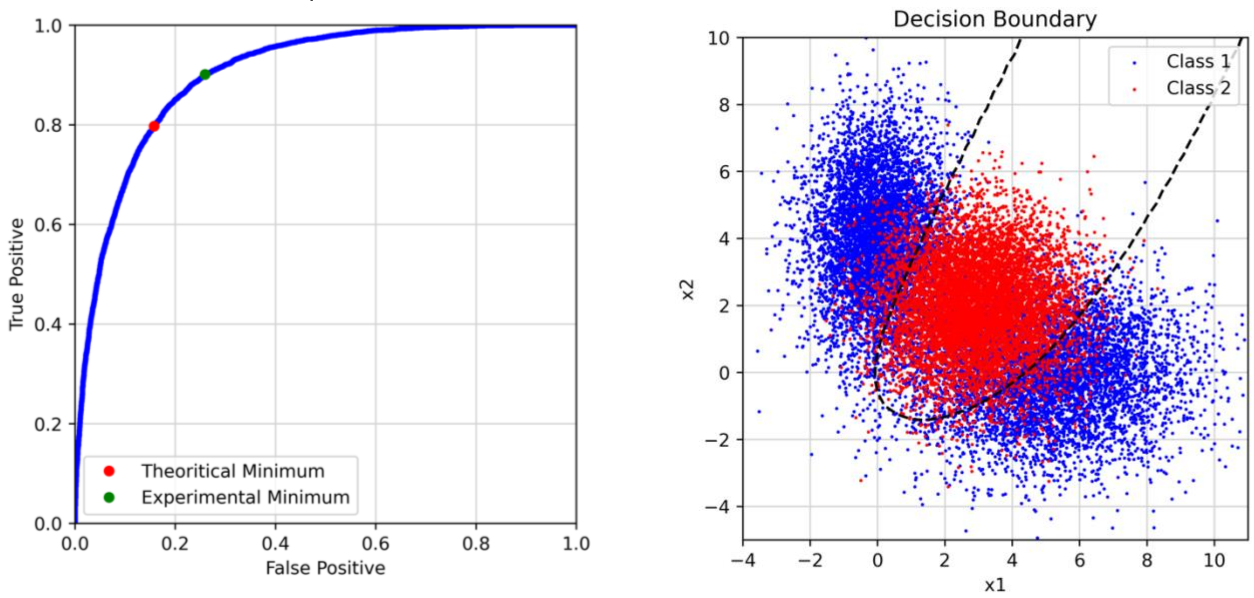


Fig: ROC Curve for Likelihood ratio test with estimated Gaussian parameters trained with 10K samples and the contour of the decision boundary.

The minimum probability of error achievable for this estimation and the threshold found empirically for that.

Table 2

	THEORETICAL	EMPERICAL
PROBABILITY OF ERROR	0.175	0.163
THRESHOLD	1.481	1.4015

The estimator has estimated the gaussian very close to the actual parameters. Hence, the classifier performs well with estimated parameters too.

Results for Training MLE with 1000 Samples

Class Label 0

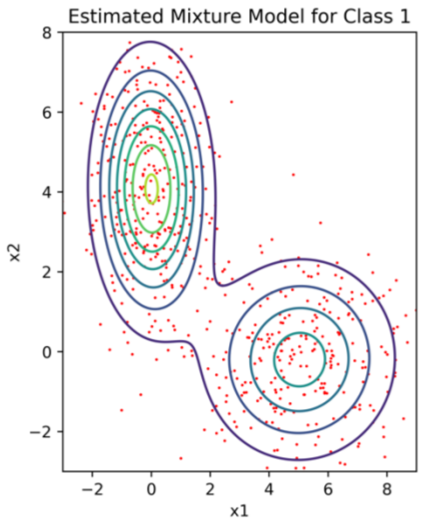
$\mathrm{M}_{01} = [5.03453564. \quad -0.19805046] \qquad \mathrm{M}_{02} = [0.00801916 \qquad 4.0794066]\ \ and$
$\mathrm{w}_1 = [0.54366140.4563386]$

$$\mathrm{C}_{01} = \begin{pmatrix} 3.50184413 & 0.04770 \\ 0.04770 & 2.11508 \end{pmatrix} \quad and\ \mathrm{C}_{02} = \begin{pmatrix} 1.18624 & -0.04206 \\ -0.04206 & 3.44799 \end{pmatrix} \quad and\ Prior\ 1\ =\ 0.604$$

Class Label 1

$\mathrm{M}_2 = [2.947300122.00819681]$

$$\mathrm{C}_2 = \begin{pmatrix} 1.7254 & 0.00788 \\ 0.00788 & 1.7004 \end{pmatrix} \quad and\ Prior\ 2\ =\ 0.396$$



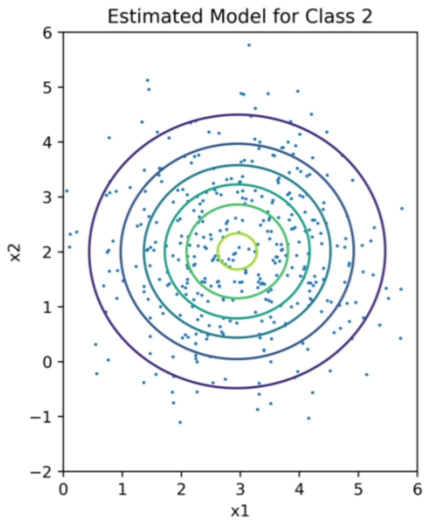


Fig: Contour plots of the estimated gaussian distributions for each class label generated by EM algorithm.

Using the estimated gaussian parameters run the Likelihood ratio classifier on the data and report the ROC Curve and Probability of error.

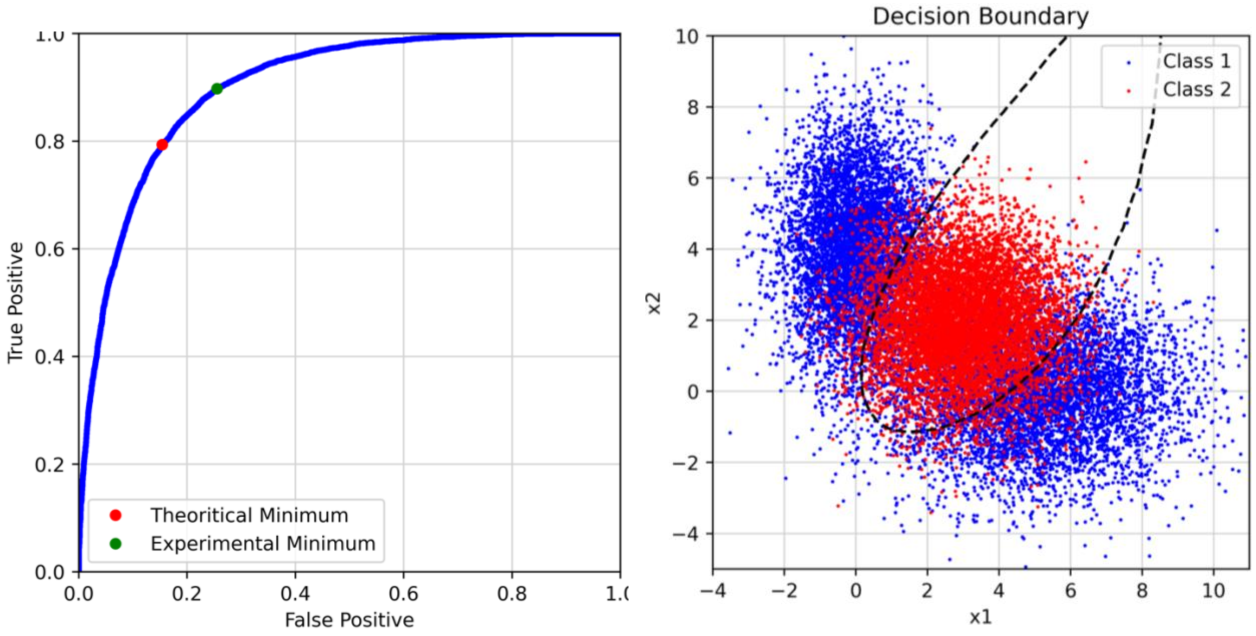


Fig: ROC Curve for Likelihood ratio test with estimated Gaussian parameters trained with 1000 samples and the contour of the decision boundary.

The minimum probability of error achievable for this estimation and the threshold found empirically for that.

Table 3

	THEORETICAL	EMPERICAL
PROBABILITY OF ERROR	0.174	0.162
THRESHOLD	1.525	1.498

The estimator wasn't as good as with 10K samples, but still performed a decent job of estimating the parameters, which in most cases might just be optimal performance to number of samples ratio.

Results for Training MLE with 100 Samples

Class Label 0

$\mathrm{M}_{01} = [5.3148 \qquad -0.2236] \qquad \mathrm{M}_{02} = [-0.2441 \qquad 4.2499]$ *and*
$\mathrm{w}_1 = [0.59400.4059]$

$$\mathrm{C}_{01} = \begin{pmatrix} 4.2822 & 0.35408 \\ 0.35408 & 1.53019 \end{pmatrix} \quad and\ \mathrm{C}_{02} = \begin{pmatrix} 0.96603 & -0.3940 \\ -0.3940 & 2.1350 \end{pmatrix} \quad and\ Prior\ 1\ =\ 0.59$$

Class Label 1

$\mathrm{M}_2 = [2.8583 \qquad 1.6583]$

$$\mathrm{C}_2 = \begin{pmatrix} 1.6211 & 0.2165 \\ 0.2165 & 1.8528 \end{pmatrix} \quad and\ Prior\ 2\ =\ 0.41$$

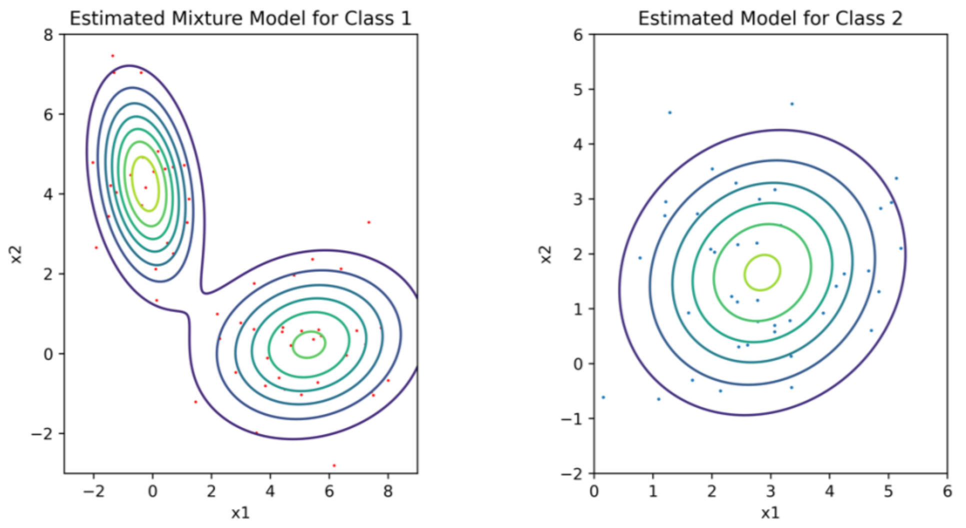
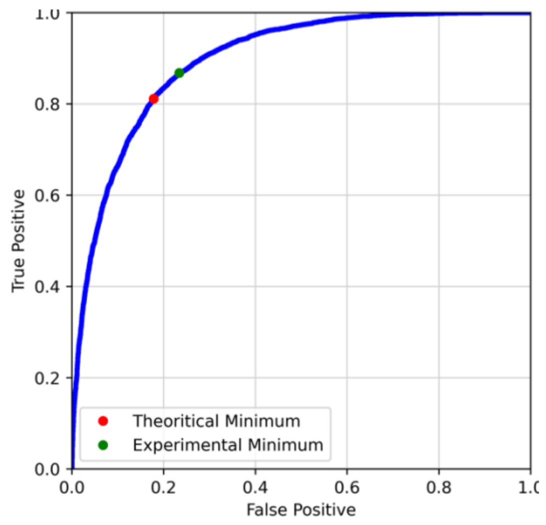


Fig: Contour plots of the estimated gaussian distributions for each class label generated by EM algorithm.

Using the estimated gaussian parameters run the Likelihood ratio classifier on the data and report the ROC Curve and Probability of error.



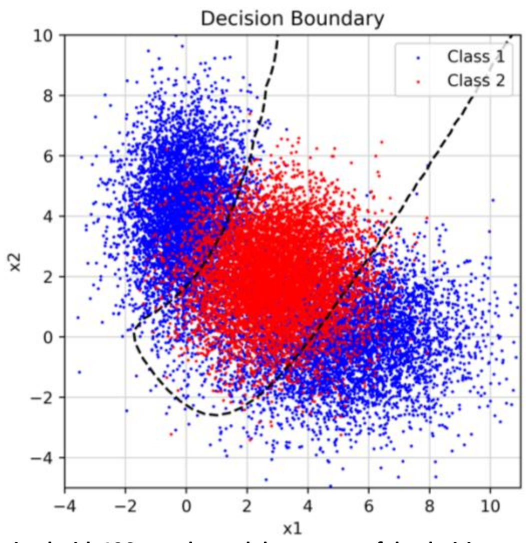


Fig: ROC Curve for Likelihood ratio test with estimated Gaussian parameters trained with 100 samples and the contour of the decision boundary.

The minimum probability of error achievable for this estimation and the threshold found empirically for that.

Table 3

	THEORETICAL	EMPERICAL
PROBABILITY OF ERROR	0.1825	0.189
THRESHOLD	1.5439	1.443

As expected, the estimator produced bad estimates for the gaussian parameters with just 100 samples. Thus, the likelihood ratio test also performs worser than with 1000 samples and with 10K samples, although it isn't that terrible with such low samples.

The Minimum P-Error doesn't increase significantly when the number of samples is reduced from 10K to 1000, it does increase when the number of samples is furthered decreased to 100.
Which shows, 10K samples is too much even though it gives a pretty accurate estimate of the gaussians, 1000 samples achieve almost similar performance with lower samples. 100 samples though is too low number of samples to achieve a reliable estimate of the gaussians.

1.PART (3) Logistic Linear Regression and Logistic Quadratic Regression

LLR Classifier

Now we train LLF classifier to approximate a linear decision boundary that transforms each input vector x via b into (1, x_1,x_2).

$$h(x,\theta) = \frac{1}{1 + e^{-w^t b(x)}}; h(x,\theta) = 1\ if\ the\ true\ label\ is\ 1\ and\ 0\ if\ label\ is\ 0$$

LLR is a linear boundary, hence the performance of LLF is going to be much worser than the likelihood ratio test.

Table 4

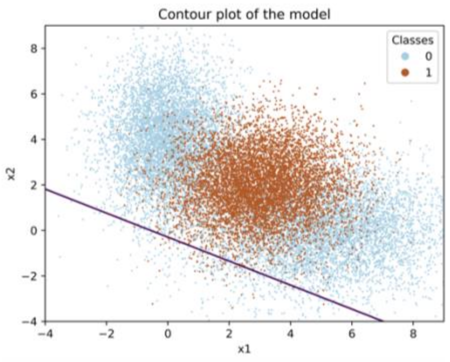


Fig: LLR Classifier trained with 10K samples.
Min P-Error = 0.40105
Threshold for Min P-Error = 0.62

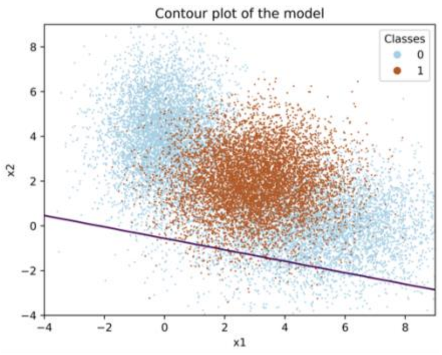


Fig: LLR Classifier trained 1000 samples
Min P-Error = 0.40105
Threshold = 0.79

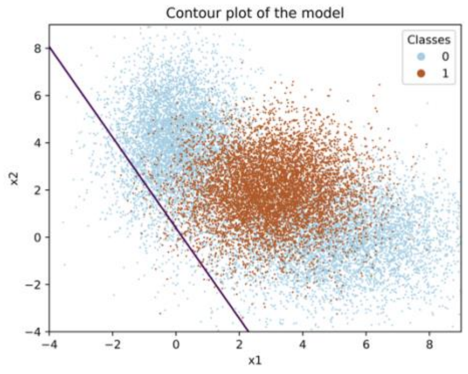


Fig: LLR Classifier trained 100 samples
Min P-Error = 0.40105
Threshold = 0.901

All lines generated in LLR are bad classifiers of the data, hence no improvement in performance despite having more samples.

LQR (OPTIONAL)

Linear logistic regression was totally inappropriate for the data set, let's try to fit quadratic regressor to the dataset to see if we perform better. Now the input vector is transformed via (1, x_1,x_2, x_1^2,x_1 * x_2,x_2^2). Since the transforming vector has quadratic terms on both axes it is capable of producing nonlinear decision boundary.

Table 5

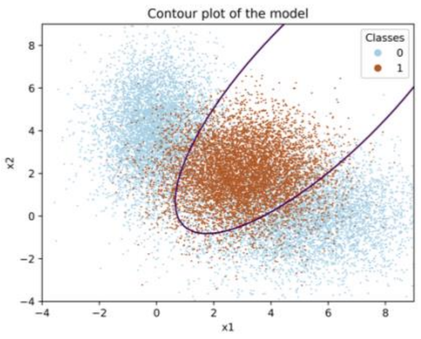


Fig: LLR Classifier trained with 10K samples.
Min P-Error = 0.175
Threshold for Min P-Error = 0.550

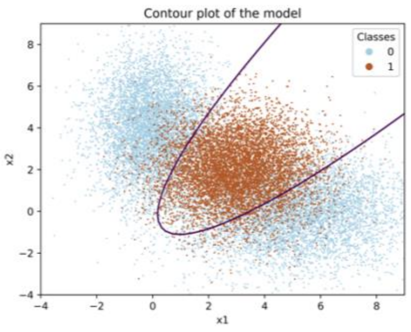


Fig: LLR Classifier trained 1000 samples
Min P-Error = 0.175
Threshold = 0.5725

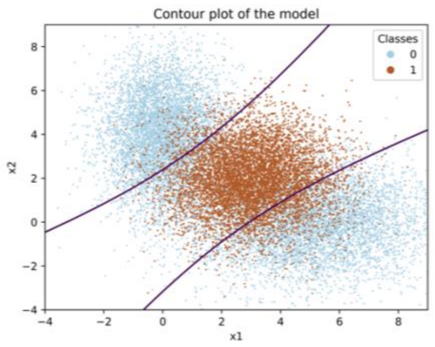


Fig: LLR Classifier trained 100 samples
Min P-Error = 0.2025
Threshold = 0.5399

2. MAP Estimation
2. Part 1) Optimization Problem

$\theta = \begin{bmatrix} x_T \\ Y_T \end{bmatrix}$ is true position of vehicle.

prior for θ.

$$p(\theta) = (2\pi\sigma_x\sigma_y)^{-1} \exp\left[\frac{-1}{2}\theta^T \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1} \theta\right]$$

Prior is a gaussian with $\mu = 0$ $\Sigma = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$

Dataset is defined as,

$$(x_i, y_i) \triangleq \underbrace{x_i = [X_i, Y_i]}_{\text{Reference point}} \text{ and } y_i = \underbrace{r_i + v_i}_{\text{Sensor observation.}}$$

y_i is composed of true reading r_i between x_i and θ_T and $v_i \sim N(0, \sigma_i^2)$.

σ_i^2 is known for each sample.

Q2) $\theta_{MAP} = \underset{\theta}{\text{argmax}} \ln P(\theta|D)$.

$\Rightarrow \underset{\theta}{\text{argmax}} \ln (P(D|\theta) + \ln P(\theta) - \ln (P(D))$

$\Rightarrow \ln P(D)$ is constant.

$\Rightarrow \underset{\theta}{\text{argmax}} \ln P(D|\theta) + \ln P(\theta)$.

$\ln P(D|\theta) = \ln \prod_{i=1}^{K} P(x_i, y_i | \theta)$.

$\Rightarrow \sum_{i=1}^{K} P(x_i, y_i | \theta)$.

$\theta_{MAP} = \underset{\theta}{\text{argmax}} \sum_{i=1}^{K} P(x_i, y_i | \theta) + \ln P(\theta)$.

From chain rule of probability.

$P(x_i, y_i | \theta) = P(y_i | x_i, \theta) + P(x_i | \theta)$.

$P(x_i)$ is independant of θ.

$P(x_i | \theta) = P(x_i)$. & $P(x_i)$ is constant w.r.t θ.

$$\underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{K} \ln P(y_i | x_i \theta) + \ln P(\theta).$$

multiply. by $-1/K$.

$$\underset{\theta}{\operatorname{argmin}} -\frac{1}{K} \sum_{i=1}^{K} \ln(P(y_i | x_i \theta)) - \frac{1}{K} \ln P(\theta).$$

$y_i = r_i + N(0, \sigma_i^2)$. $\quad r_i = \|x_i - \theta\|$

$$P(y_i | x_i \theta) = g(y_i, x_i, \sigma_i^2) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{-1}{2\sigma_i^2} (y_i - r_i^2)^2\right)$$

$$\ln P(y_i | x_i \theta) = \ln\left(\frac{1}{\sqrt{2\pi\sigma_i^2}}\right) + \left(\frac{-1}{2\sigma_i^2} (y_i - r_i^2)^2\right)$$

$$\Rightarrow \underset{\theta}{\operatorname{argmin}} \frac{1}{2K} \sum_{i=1}^{K} \ln(2\pi\sigma_i^2) + \frac{1}{\sigma_i^2} (y_i - r_i)^2 - \frac{1}{K} \ln P(\theta)$$

$$\Rightarrow \underset{\theta}{\operatorname{argmin}} \underbrace{\frac{1}{2K} \sum_{i=1}^{K} \ln(2\pi\sigma_i^2)}_{\text{constant}} + \frac{1}{2K} \sum_{i=1}^{K} \frac{(y_i - r_i)^2}{\sigma_i^2} - \frac{1}{K} \ln P(\theta).$$

$$\Rightarrow \underset{\theta}{\operatorname{argmin}} \frac{1}{2K} \sum_{i=1}^{K} \sigma_i^{-2} (y_i - r_i)^2 - \frac{1}{K} \ln P(\theta).$$

$P(\theta) = g(0, \sigma)$.

$$\ln P(\theta) = -\ln 2\pi\sigma_x\sigma_y - \frac{1}{2} \theta^T \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1} \theta.$$

$$\underset{\theta}{\operatorname{argmin}} \frac{1}{2K} \sum_{i=1}^{K} \sigma_i^{-2} (y_i - r_i)^2 + \underbrace{\frac{1}{K} \ln 2\pi\sigma_x\sigma_y}_{\text{constant}} + \frac{1}{2K} \theta^T \begin{bmatrix} \sigma_x & 0 \\ 0 & \sigma_y \end{bmatrix}^{-1} \theta$$

$$\theta_{MAP} = \frac{1}{2K} \sum_{i=1}^{K} \sigma_i^{-2} \left(y_i - \|x_i - \theta\|\right)^2 + \frac{1}{2K} \theta^T \begin{bmatrix} \sigma_x & 0 \\ 0 & \sigma_y \end{bmatrix}^{-1} \theta$$

2.PART 2) CODE WALKTHROUGH

The code was tested for situations with K = 1,2,3,4 and 40.
True Point condition r ≤ ¾ evenly spaced in the circle by angle.
Measurements taken with noise 0.3 ignoring negatives.

1. We initialize the true position in the code first according to the condition.
2. Run a loop of the entire functionality, simulating conditions K = 1,2,3,4 and 40
3. For each K, compute the reference points via linspace function in python such that they are evenly spaced somewhere inside the unit circle.
4. For all i = [1,K]
    a. Init $\sigma_i = 0.3$
    b. Init $x_i = (\cos\theta i, \sin\theta i)$
    c. Calculate y_i as $|| x_i - \text{truepos} || + N(0, \sigma_i^2)$
    d. Repeat c if $y_i < 0$.
5. Define the MAP_Loglikelihood function as derived above.
6. Plot the contours of the function using matplotlib contour and the minimum of the MAP function as the MAP estimate of the position of the vehicle with reference points.

As the number of reference points increasing, it is evident from the output of the code that the MAP performs better and gets closer to the True Position of the robot.
Hence, asymptotically($K \to \infty$), the MAP estimate becomes our MLE estimate.

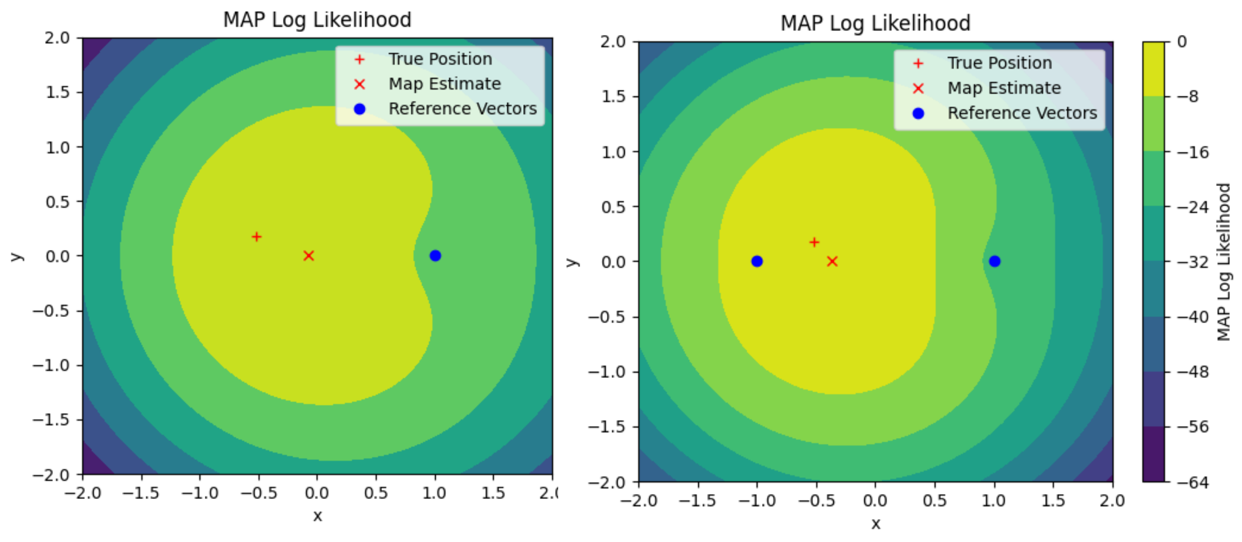


Fig: Contour plot of the likelihood function along with reference points and true position

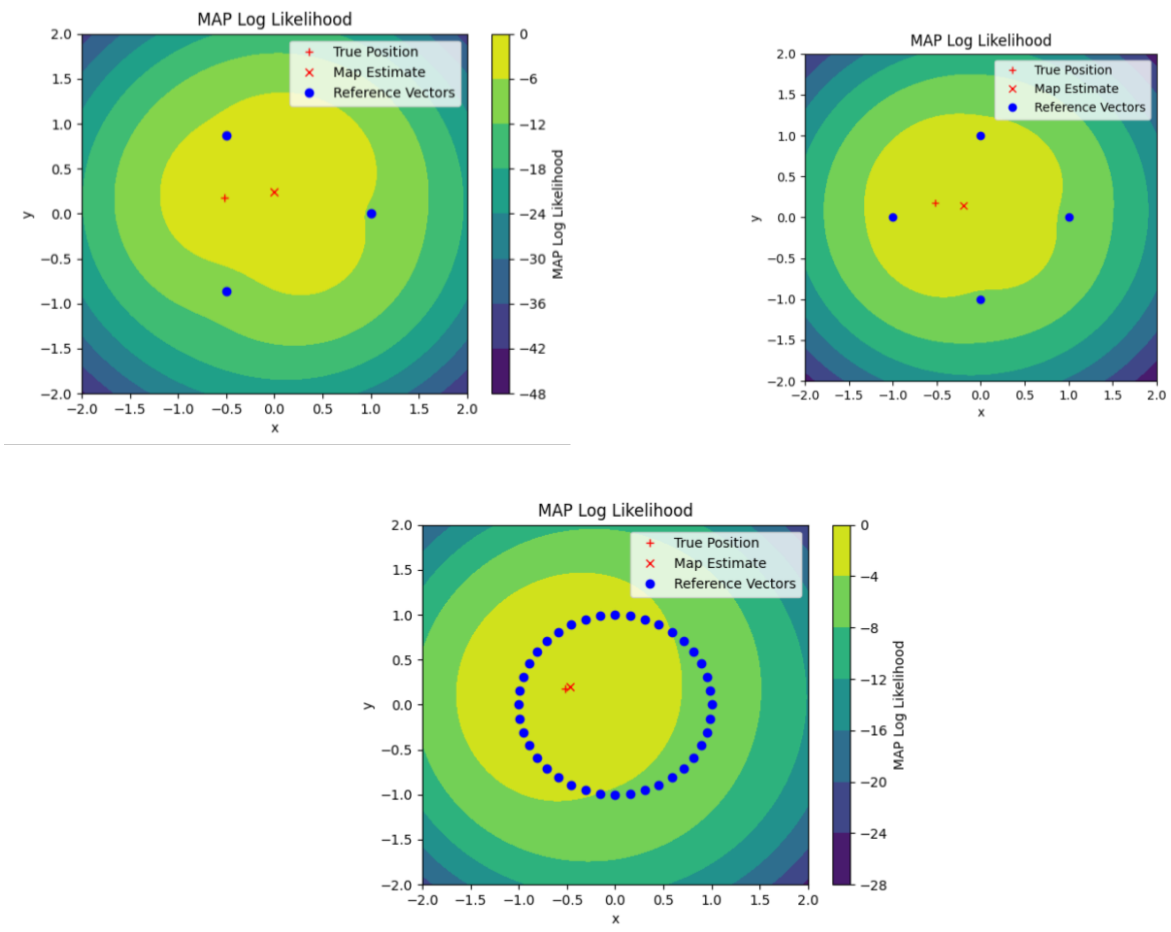


Fig: Contour plot of the likelihood function along with reference points and true position

It can be seen from the plots that the MAP Estimate of the position of vehicle gets closer and closer as the number of reference vectors increase. And at K = 40, it is no more dependent on the prior and pretty accurate.

3. Part 1) Risk classifier

Given,

$$\lambda(\alpha_i | \omega_j) = \begin{cases} 0 , & i = j \quad i,j = 1 \dots c \text{ (correct)} \\ \lambda_r , & i = c+1 \text{ (mismatch)} \\ \lambda_s , & \text{otherwise (reject)} \end{cases}$$

$$R(D = i | x) = \underbrace{\sum_{i=1}^{c} \lambda(i|l) p(L = l | x)}_{= 1}$$

$$\lambda(i|l) = \lambda_s \quad \text{for } i \neq l$$

$$\therefore R(D = i | x) = \lambda_s (1 - P(L = i | x))$$

Find minimum risk over all classes.

$$\min_{i = 1, 2 \dots c} \lambda_s (1 - P(L = i | x)) \Rightarrow \min_{i = 1, 2 \dots c} P(L = i | x)$$

a) assume the maximum as y_i.

min risk $\rightarrow P(L = y_i | x) > P(L = j | x)$.

Risk with deciding $c+1$,

$$R(D = c+1 | x) = \sum_{i=1}^{c} \lambda(c+1 | l) p(L = l | x)$$

We know, $\lambda(c+1|L) = \lambda_r$ & $\sum_{i=1}^{c} P(L=1|x) = 1$

$$\lambda_r \times 1 = \lambda_r.$$

Finally, $R(D=y_i|x) > R(D=c+1|x)$ to calculate the decision boundary.

$$\lambda_s(1-P(L=i|x)) > \lambda_r$$

$$\Rightarrow -P(L=y_i|x) > \frac{\lambda_r}{\lambda_s} - 1$$

$$\boxed{P(L=y_i|x) < 1 - \frac{\lambda_r}{\lambda_s}}$$ rejection criteria.

---

if $\lambda_r = 0$, there is 0 penalty for rejection, there is no risk for mismatch.

Always reject (choose $L=c+1$)

$$P(L = y_i \mid x) < 1 - \frac{\lambda_r}{\lambda_s} \rightarrow 0$$

$$\boxed{\therefore P(L = y_i \mid x) < 1}$$

which is always true since probability is always less than 1.

if $P(L = y_i \mid x) = 1 \rightarrow$ 100% certain of the current label.

then no risk, classifier always certain.

---

if mismatch > reject. $\lambda_s > \lambda_r$.

it is always better to guess than reject.

When guessing.

$$\text{penalty} = \begin{cases} 0, & \text{if } i = j \\ \lambda_s, & i \neq j \end{cases}$$

but $\lambda_s < \lambda_r$. $\max(0, \lambda_s) < \lambda_r$.

thus classifier should always decide $L \in 1 \ldots c$.

$$P(L = y_i | x) < 1 - \frac{\lambda_r}{\lambda_s}$$

$$\Rightarrow \underbrace{\frac{\lambda_r}{\lambda_s} > 1}_{\text{from assumption}} \Rightarrow 1 - \frac{\lambda_r}{\lambda_s} < 0.$$

hence,

$$P(L = y_i | x) < 1 - \frac{\lambda_r}{\lambda_s} < 0.$$

but, $P(L = y_i | x) \geq 0$ by definition of probability.

Rejection criteria will never be met.