Helena Calatrava

November 2022

EECE5644 | Homework 3

Please submit your solutions on Canvas in a single PDF file that includes all math, numerical and visual results. Either include a link to your code in an online repository or include the code as an appendix in the PDF file. The code is not graded, but helps verify your results are feasible as claimed. Only results and discussion presented in the PDF will be graded, so do not link to an external location where further results may be presented.

This is a graded assignment and the entirety of your submission must contain only your own work. You may benefit from publicly available literature including software (not from classmates), as long as these sources are properly acknowledged in your submission. All discussions and materials shared during office periods are also acceptable resources and these tend to be very useful, so participate in office periods or take a look at their recordings. Cite your sources as appropriate.

By submitting a PDF file in response to this take home assignment you are declaring that the contents of your submission, and the associated code is your own work, except as noted in your citations to resources.

Please, find the code in the following Drive folder:
https://drive.google.com/drive/folders/1Xmewr_a4lS-LqvwKeM-fHitm9s5GpeXF?usp=sharing

(or check this github repository: https://github.com/hcalatrava/EECE5644_HW3)

Question 1 (50%)

The probability density function (pdf) for a 2-dimensional real-valued random vector $\mathbf{X}$ is as follows: $p(\mathbf{x}) = p(\mathbf{x}|L=0)P(L=0) + p(\mathbf{x}|L=1)P(L=1)$. Here L is the true class label that indicates which class-label-conditioned pdf generates the data.

The class priors are $P(L=0) = 0.6$ and $P(L=1) = 0.4$. The class class-conditional pdfs are $p(\mathbf{x}|L=0) = w_1 g(\mathbf{x}|\mathbf{m}_{01}, \mathbf{C}_{01}) + w_2 g(\mathbf{x}|\mathbf{m}_{02}, \mathbf{C}_{02})$ and $p(\mathbf{x}|L=1) = g(\mathbf{x}|\mathbf{m}_1, \mathbf{C}_1)$, where $g(\mathbf{x}|\mathbf{m}, \mathbf{C})$ is a multivariate Gaussian probability density function with mean vector $\mathbf{m}$ and covariance matrix $\mathbf{C}$. The parameters of the class-conditional Gaussian pdfs are: $w_1 = w_2 = 1/2$, and

$$\mathbf{m}_{01} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \quad \mathbf{C}_{01} = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \quad \mathbf{m}_{02} = \begin{bmatrix} 0 \\ 4 \end{bmatrix} \quad \mathbf{C}_{02} = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \quad \mathbf{m}_1 = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \quad \mathbf{C}_1 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

For numerical results requested below, generate the following independent datasets each consisting of iid samples from the specified data distribution, and in each dataset make sure to include the true class label for each sample. Save the data and use the same data set in all subsequent exercises.

- D_{train}^{100} consists of 100 samples and their labels for training;
- D_{train}^{1000} consists of 1000 samples and their labels for training;
- D_{train}^{10000} consists of 10000 samples and their labels for training;
- $D_{validate}^{20K}$ consists of 20000 samples and their labels for validation;

Part 1: (10%) Determine the theoretically optimal classifier that achieves minimum probability of error using the knowledge of the true pdf. Specify the classifier mathematically and implement it; then apply it to all samples in $D_{validate}^{20K}$. From the decision results and true labels for this validation set, estimate and plot the ROC curve of this min-P(error) classifier, and on the ROC curve indicate, with a special marker, indicate the point that corresponds to the min-P(error) classifier's operating point. Also report your estimate of the min-P(error) achievable, based on counts of decision-truth label pairs on $D_{validate}^{20K}$. Optional: As supplementary visualization, generate a plot of the decision boundary of this classification rule overlaid on the validation dataset. This establishes an aspirational performance level on this data for the following approximations.

Part 2: (20%) (a) Using the maximum likelihood parameter estimation technique, estimate the class priors and class conditional pdfs using training data in D_{train}^{10000}. As class conditional pdf models, for $L=0$ use a Gaussian Mixture model with 2 components, and for $L=1$ use a single Gaussian pdf model. For each estimated parameter, specify the maximum-likelihood-estimation objective function that is maximized as well as the iterative numerical optimization procedure used, or if applicable, for analytically tractable parameter estimates, specify the estimator formula. Using these estimated class priors and pdfs, design and implement an approximation of the min-P(error) classifier, apply it on the validation dataset $D_{validate}^{20K}$. Report the ROC curve and minimum probability of error achieved on the validation dataset with this classifier that is trained with 10000 samples. (b) Repeat Part (2a) using D_{train}^{1000} as the training dataset. (c) Repeat Part (2a) using D_{train}^{100} as the training dataset. How does the performance of your approximate min-P(error) classifier change as the model parameters are estimated (trained) using fewer samples?

Part 3: (20%) (a) Using the maximum likelihood parameter estimation technique train a logistic-linear-function-based approximation of class label posterior function given a sample. As in part 2, repeat the training process for each of the three training sets to see the effect of training set sample count; use the validation set for performance assessment in each case. When optimizing the

parameters, specify the optimization problem as minimization of the negative-log-likelihood of the training dataset, and use your favorite numerical optimization approach, such as gradient descent or Matlab's *fminsearch* or Python's *minimize*. Use the trained class-label-posterior approximations to classify validation samples to approximate the minimum-P(error) classification rule; estimate the probability of error that these three classifiers attain using counts of decisions on the validation set. Optional: As supplementary visualization, generate plots of the decision boundaries of these trained classifiers superimposed on their respective training datasets and the validation dataset. (b) Repeat the process described in Part (3a) using a logistic-quadratic-function-based approximation of class label posterior functions given a sample.

Note 1: With $\mathbf{x}$ representing the input sample vector and $\mathbf{w}$ denoting the model parameter vector, logistic-linear-function refers to $h(\mathbf{x},\mathbf{w}) = 1/(1+e^{-\mathbf{w}^{\mathrm{T}}\mathbf{z}(\mathbf{x})})$, where $\mathbf{z}(\mathbf{x}) = [1,\mathbf{x}^{\mathrm{T}}]^{\mathrm{T}}$; and logistic-quadratic-function refers to $h(\mathbf{x},\mathbf{w}) = 1/(1+e^{-\mathbf{w}^{\mathrm{T}}\mathbf{z}(\mathbf{x})})$, where $\mathbf{z}(\mathbf{x}) = [1,x_1,x_2,x_1^2,x_1x_2,x_2^2]^{\mathrm{T}}$.

Hint: The classifier designed in Part 1 uses the true pdf knowledge and achieves theoretically optimum performance in terms of minimizing P(error). The classifiers designed in Part 2 are approximations of the theoretically optimum classification rule using the correct functional form of the data pdf, however the quality of approximation for these generative models will get worse as their parameters are estimated with fewer training samples. The classifiers in Part 3 attempt to directly approximate class label posteriors, and the approximtion capability increases as the model gets more complex (linear to quadratic). The classifiers in Part 3, however, are limited by the approximation capability of their functional form. While the classifiers in Part 2 can asymptotically approach the performance level of the theoretically optimal one if they are trained with more data, the classifiers in Part 3 are bounded in performance by the fact that they can only generate linear or quadratic decision boundaries, so no amount of training data will enable them to asymptotically approximate the theoretically optimal classification rule (which has a classification boundary that is more complex than quadratic).

Question 1 - Part 1

Theoretical optimal classifier that achieves min. Pe with knowledge of the pdfs → Bayes Classifier.

Class-conditional pdf ratio:

True Label L_i, Decision $D_i \in \{0,1\}$

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

$$g(\underline{x}|\underline{m},\underline{C}) = \frac{1}{\sqrt{2\pi|C|}} \exp\left(-\frac{1}{2}(\underline{x}-\underline{m})^T \underline{C}^{-1}(\underline{x}-\underline{m})\right)$$

$$p(\underline{x}) = p(L=0)\cdot p(\underline{x}|L=0) + p(L=1)\cdot p(\underline{x}|L=1)$$

$\lambda_{ij} = \lambda(D=i, L=j)$

$$R(D=i|\underline{x}) = \lambda_{ii}\, p(L=i|\underline{x}) + \lambda_{ij}\, p(L=j|\underline{x})$$

$R(D=i|\underline{x}) > R(D=j|\underline{x}) \rightarrow$ we decide $L=j$

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

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

NOTE: Bayes Rule $p(L=i|\underline{x}) = \frac{p(\underline{x}|L=i)\, p(L=i)}{p(\underline{x})}$

(*) ↓

$$\frac{p(\underline{x}|L=0)}{p(\underline{x}|L=1)} < \frac{p(L=1)}{p(L=0)} \frac{(\lambda_{11}-\lambda_{01})}{(\lambda_{00}-\lambda_{10})}$$

$\cdot(-1)$ ↓

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

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

Classifier Mathematical Expression

minimum risk solution (0-1 loss)

If we make the minimum risk decision for each individual sample, we are minimizing the expected overall risk. Applying the Bayes rule we see that the theoretically optimal threshold can be expressed as a function of the likelihood ratio and the class prior probabilities. The theoretically optimal threshold value that achieves minimum P(error) is obtained by assigning zero-loss to correct decisions and 1-loss to incorrect decisions (0-1 loss). The theoretical threshold in this case, considering the knowledge of class prior probabilities, is calculated as

$$\gamma_{opt,\,theory} = \frac{p(L=0)}{p(L=1)} = \frac{0.6}{0.4} = 1.5$$

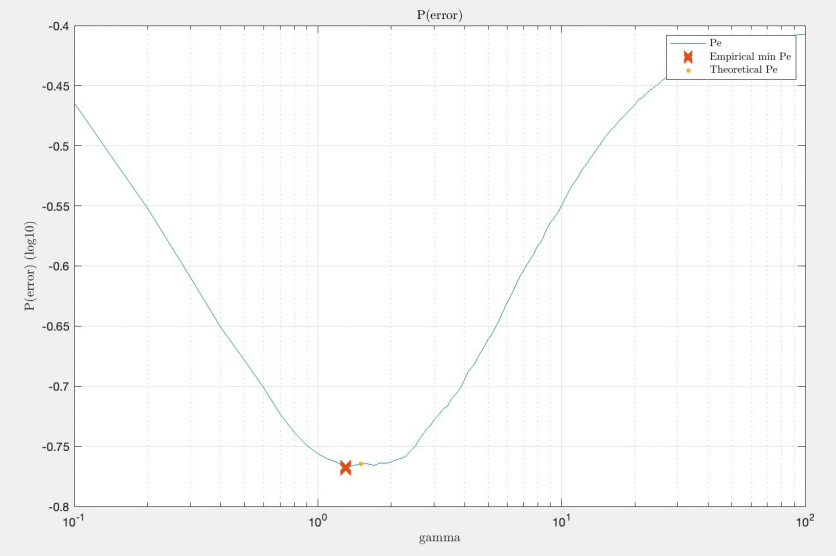
Also, the optimal threshold (i.e. classifier operating point) is empirically estimated with MATLAB simulations. Different values of the threshold γ are used to compute the probability of error, and we can see how the empirically found threshold giving the lowest Pe corresponds to the ratio of class prior probabilities (matching the theoretically obtained threshold).

→ Comparison of empirical and theoretical optimal points:

	Threshold	min Pe
Theoretical	1,5	0,1719
Empirical	1,3013	0,1708

The theoretical and empirical minimum probabilities of error are very close to each other (as seen both in the table and figure), thus suggesting that the findings are correct. The number of equally spaced tested values of threshold γ is 1e3, between 0 and 1e2.

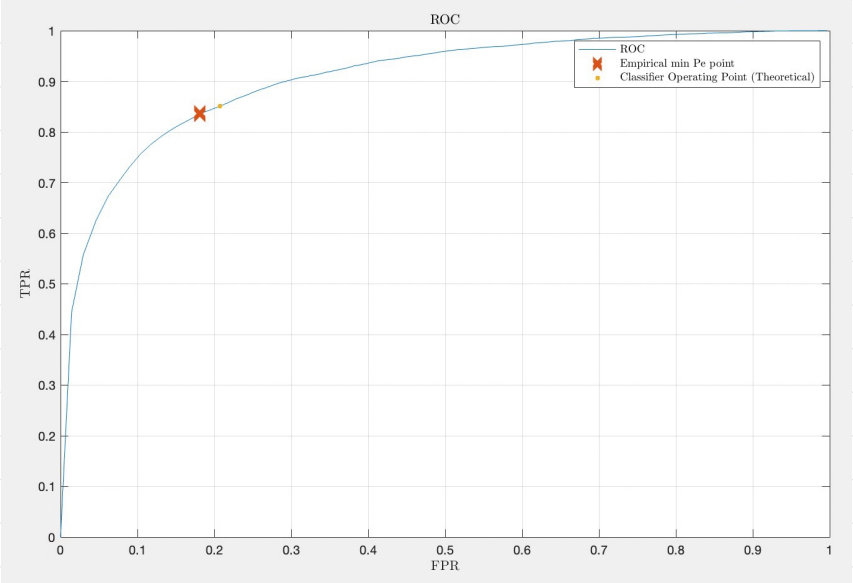
→ P(error) plot



The probability of error is plotted in logarithmic scale for a clearer visualization.

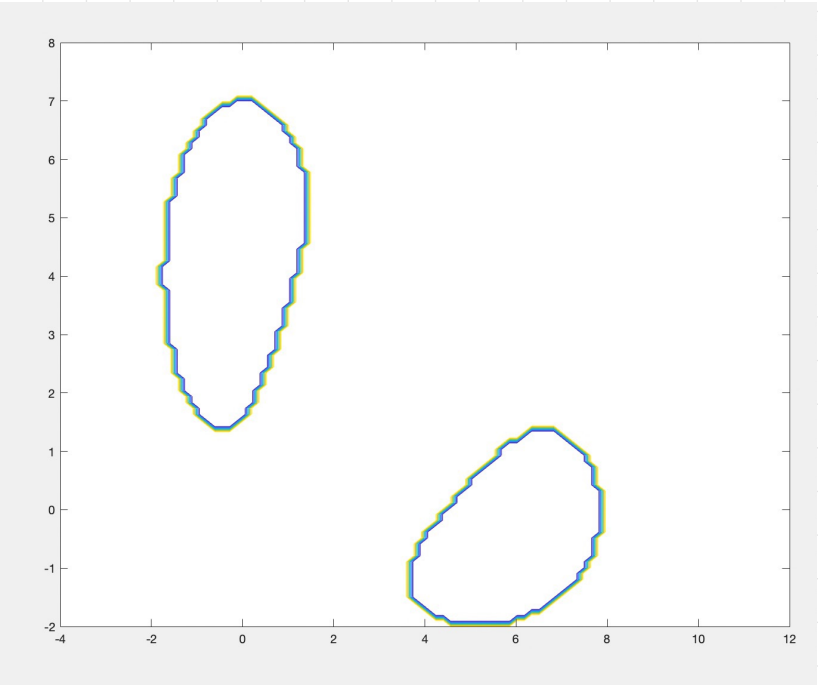
What do we mean by theoretical and empirical?
We have seen with the mathematical derivation that the theoretical threshold is the ratio of the prior probabilities, which in this case is 1.5, and corresponds to the **theoretical classifier operating point**. However, when implementing the classifier, we do not (empirically) find the exact same value of threshold (gamma) giving the minimum probability of error. I have tried with several Matlab seeds and I see that the empirical threshold giving minimum Pe is netween 1.3 and 1.7. This is giving the randomness of the generated data, as when changing the seed we can get a different value of operating point threshold and also of minimum Pe. Also, this is why we see that the minimum achievable Pe in our classifier (implemented one) can differ from the one obtained for a gamma very close to 1.5. The theoretical Pe has been obtained by checking which is the obtained Pe in the implemented classifier when using the closest threshold to 1.5.

→ ROC curve



The shown ROC curve was obtained by applying the previously described (theoretically) classifier to the validation dataset (20k samples) and for the amount of threshold γ values specified before, between 0 and 1e2.

Decision boundary: As explained in part 3, the theoretically optimal classification rule has a classification boundary that is more complex than quadratic.



(optional):

This would make sense as a boundary because it separates GMM data (class 0) from non-GMM data (class 1).

Hope it's correct!

Question 1- Part 2

With MLE, estimation of the class priors and class conditional PDFs.

→ **Class prior estimation**

$p(L=0)$, $p(L=1)$ } we compute the priors with the class labels from the TRAINING SET, which are given to us.

→ **Class conditional pdfs estimation**

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

parameter vector $\underline{\theta}$, which contains the mean $\underline{m}_1$ and covariance $\underline{\underline{C}}_1$ → $\underline{\theta} = [\underline{m}_1, \underline{\underline{C}}_1]$

TRAINING SET

$$p(\underline{X}|\underline{\theta}) = \prod_{k=1}^{n} p(\underline{x}_k|\underline{\theta})$$

($\underline{x}_k$: Kth two-dim sample; $\underline{\theta}$ ≡ parameter vector; assumption: i.i.d. samples)

→ With the MLE we find the value of $\underline{\theta}$ that maximizes $p(\underline{X}|\underline{\theta})$

log-likelihood function: $l(\underline{\theta}) = \ln p(\underline{X}|\underline{\theta}) \Rightarrow \hat{\underline{\theta}} = \underset{\underline{\theta}}{\operatorname{argmax}}\, l(\underline{\theta})$

(implicit dependence on our training set $\underline{X}$)

$$\nabla_{\underline{\theta}} l = \nabla_{\underline{\theta}} \ln p(\underline{X}, \underline{\theta}) = \sum_{k=1}^{n} \nabla_{\underline{\theta}} \ln p(\underline{x}_k|\underline{\theta})$$

$\boxed{\nabla_{\underline{\theta}} l = 0}$ → gives us p equations (number of elements in the parameter vector $\underline{\theta}$)

$$\nabla_{\underline{\theta}} \equiv \left[\frac{\partial}{\partial\theta_1} \cdots \frac{\partial}{\partial\theta_p}\right]$$

number of samples in training set

$$\ln p(\underline{x}_k|\underline{\theta}) = -\frac{1}{2}\ln\left[(2\pi)^{d}|\underline{\underline{C}}|\right] - \frac{1}{2}(\underline{x}_k - \underline{m})^T \underline{\underline{C}}^{-1}(\underline{x}_k - \underline{m})$$

$$\nabla_{\underline{\theta}} \ln p(\underline{x}_k|\underline{\theta}) = \begin{bmatrix} \frac{\partial}{\partial \underline{m}} \ln p(\underline{x}_k|\underline{\theta}) \\ \frac{\partial}{\partial \underline{\underline{C}}} \ln p(\underline{x}_k|\underline{\theta}) \end{bmatrix}; \quad \boxed{\nabla_{\underline{\theta}} \ln p(\underline{X}, \underline{\theta}) = \begin{bmatrix} \sum_{k=1}^{n} \frac{\partial}{\partial \underline{m}} \ln p(\underline{x}_k|\underline{\theta}) \\ \sum_{k=1}^{n} \frac{\partial}{\partial \underline{\underline{C}}} \ln p(\underline{x}_k|\underline{\theta}) \end{bmatrix}}$$

→ $$\frac{\partial}{\partial \underline{m}} \ln p(\underline{x}_k|\underline{\theta}) = -\frac{1}{2}\frac{\partial}{\partial \underline{m}}(\underline{x}_k - \underline{m})^T \underline{\underline{C}}^{-1}(\underline{x}_k - \underline{m}) = -\frac{1}{2}(-2)\cdot \underline{\underline{C}}^{-1}(\underline{x}_k - \underline{m}) = \underline{\underline{C}}^{-1}(\underline{x}_k - \underline{m})$$

$$\sum_{k=1}^{n} \frac{\partial}{\partial \underline{m}} \ln p(\underline{x}_k|\underline{\theta}) = \sum_{k=1}^{n} \underline{\underline{C}}^{-1}(\underline{x}_k - \underline{m})$$

→ $$\sum_{k=1}^{n} \frac{\partial}{\partial \underline{\underline{C}}} \ln p(\underline{x}_k|\underline{\theta}) = \frac{\partial}{\partial \underline{\underline{C}}} \sum_{k=1}^{n} \ln p(\underline{x}_k|\underline{\theta}) = \frac{\partial}{\partial \underline{\underline{C}}}\left[\underbrace{\text{constant}}_{\beta} - \frac{N}{2}\ln|\underline{\underline{C}}| - \frac{1}{2}\sum_{k=1}^{n}(\underline{x}_k - \underline{m})^T \underline{\underline{C}}^{-1}(\underline{x}_k - \underline{m})\right] =$$

$$= \frac{\partial}{\partial \underline{\underline{C}}}\left[\beta + \frac{N}{2}\ln|\underline{\underline{C}}^{-1}| - \frac{1}{2}\sum_{k=1}^{n} TR\left[(\underline{x}_k - \underline{m})^T \underline{\underline{C}}^{-1}(\underline{x}_k - \underline{m})\right]\right] =$$

$$= \frac{\partial}{\partial \underline{\underline{C}}}\left[\beta + \frac{N}{2}\ln|\underline{\underline{C}}^{-1}| - \frac{1}{2}\sum_{k=1}^{n} TR\left[(\underline{x}_k - \underline{m})^T (\underline{x}_k - \underline{m})\underline{\underline{C}}^{-1}\right]\right] = \circledast$$

$$(*) = \frac{N}{2}\underline{\underline{C}}^{-1} - \frac{1}{2}\sum_{K=1}^{n}(\underline{x}_K - \underline{\mu})(\underline{x}_K - \underline{\mu})^T = \sum_{K=1}^{n}\frac{\partial}{\partial \underline{\underline{C}}}\ln p(\underline{x}_K|\underline{\theta})$$

Properties used for the derivative:

1) $\frac{\partial}{\partial A}\log|A| = A^{-T} = (A^{-1})^T$

$$\frac{\partial}{\partial \underline{\underline{C}}^{-1}}\left[\frac{N}{2}\ln|\underline{\underline{C}}^{-1}|\right] = \frac{N}{2}\underline{\underline{C}}^T = \frac{N}{2}\underline{\underline{C}} \quad \left(\text{covariance matrix, i.i.d. } \underline{\underline{C}}^T = \underline{\underline{C}}\right)$$

2) $\frac{\partial}{\partial \underline{\underline{A}}}TR[\underline{\underline{A}}\,\underline{\underline{B}}] = \frac{\partial}{\partial \underline{\underline{A}}}TR[\underline{\underline{B}}\,\underline{\underline{A}}] = \underline{\underline{B}}^T$

Now we set these equations to zero:

Equation 1 - $\hat{\underline{\mu}}$

$$\sum_{K=1}^{n}\frac{\partial}{\partial \underline{\mu}}\ln p(\underline{x}_K|\underline{\theta}) = \sum_{K=1}^{n}\underline{\underline{C}}^{-1}(\underline{x}_K - \underline{\mu}) = 0 \rightarrow \quad (1)\quad \boxed{\hat{\underline{\mu}} = \frac{1}{N}\sum_{K=1}^{n}\underline{x}_K}$$

The ML estimate for the mean vector is the SAMPLE MEAN.

↓

We are expecting that the accuracy of this estimator increases with N.

Equation 2 - $\hat{\underline{\underline{C}}}$

$$\sum_{K=1}^{n}\frac{\partial}{\partial \underline{\underline{C}}}\ln p(\underline{x}_K|\underline{\theta}) = \frac{N}{2}\underline{\underline{C}} - \frac{1}{2}\sum_{K=1}^{n}(\underline{x}_K - \underline{\mu})(\underline{x}_K - \underline{\mu})^T = 0 \rightarrow \quad (2)\quad \boxed{\hat{\underline{\underline{C}}} = \frac{1}{N}\sum_{K=1}^{n}(\underline{x}_K - \hat{\underline{\mu}})(\underline{x}_K - \hat{\underline{\mu}})^T}$$

The ML estimate for the covariance matrix is the arithmetic average of the N matrices $(\underline{x}_K - \hat{\underline{\mu}})(\underline{x}_K - \hat{\underline{\mu}})^T$.

Therefore, we estimate $\underline{\mu}$ and $\underline{\underline{C}}$ for classes 1 as indicated in (1) and (2). However in the case of class 0, as we have a GMM, we need to estimate the parameters iteratively, using the ITERATIVE EXPECTATION-MAXIMIZATION (EM) algorithm.

For this task, I am using the **fitgmdist MATLAB function**, which returns a Gaussian mixture distribution model with K components. We set K to 2, as we known that we have 2 components in the mixture. This function:

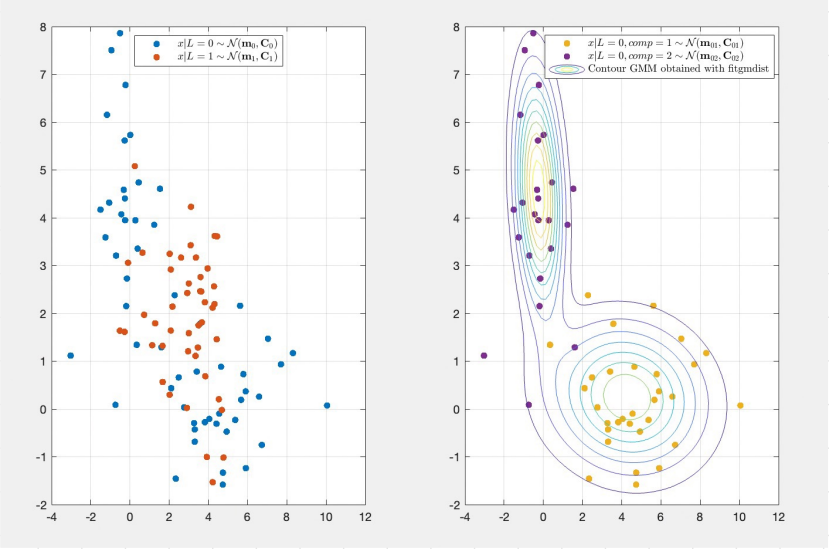
→ Implements the **K-means** algorithm for initialization to choose K=3 initial cluster centers.

→ Uses the **EM algorithm** to iteratively solve the estimation problem. The following steps are repeated until convergence:

a. Fix the parameters and solve the posterior distribution for the hidden variables.

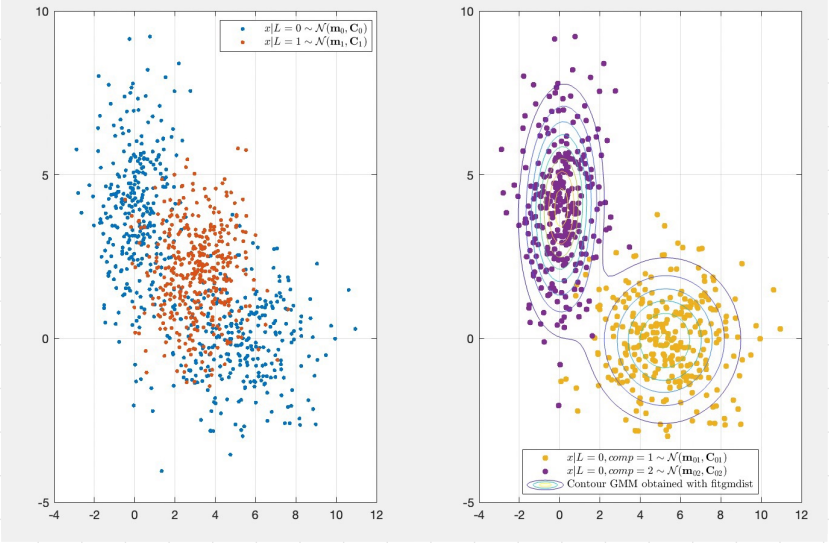
b. Fix the posterior distribution for the hidden variables and optimize the parameters using the expected values of the hidden variables.

After fitting the data of the GMM, we get :

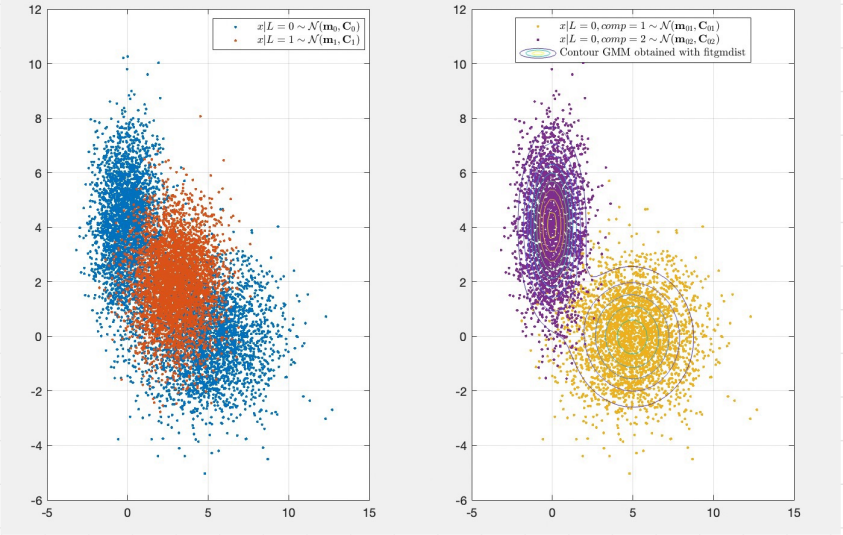


N ≡ number of training samples

N = 100



N = 1e3



N = 10e3

We can see how the selected MATLAB function to fit the GMM data makes possible to find the parameters of the GMM model for the three training datasets.

→ Classifier min P(error)

In this case, to compute the likelihood ratio and the class prior ratio that is required for the Bayesian classifier, we use the values obtained with the MLE and iterative EM estimation.

As in the previous part of this question, the min-P(error) classifier is the Bayes classifier with minimum risk → Same as previous question but without perfect knowledge of $\underline{m}_1$, $\underline{m}_{01}$, $\underline{m}_{02}$, $\underline{\underline{C}}_1$, $\underline{\underline{C}}_{01}$, $\underline{\underline{C}}_{02}$.

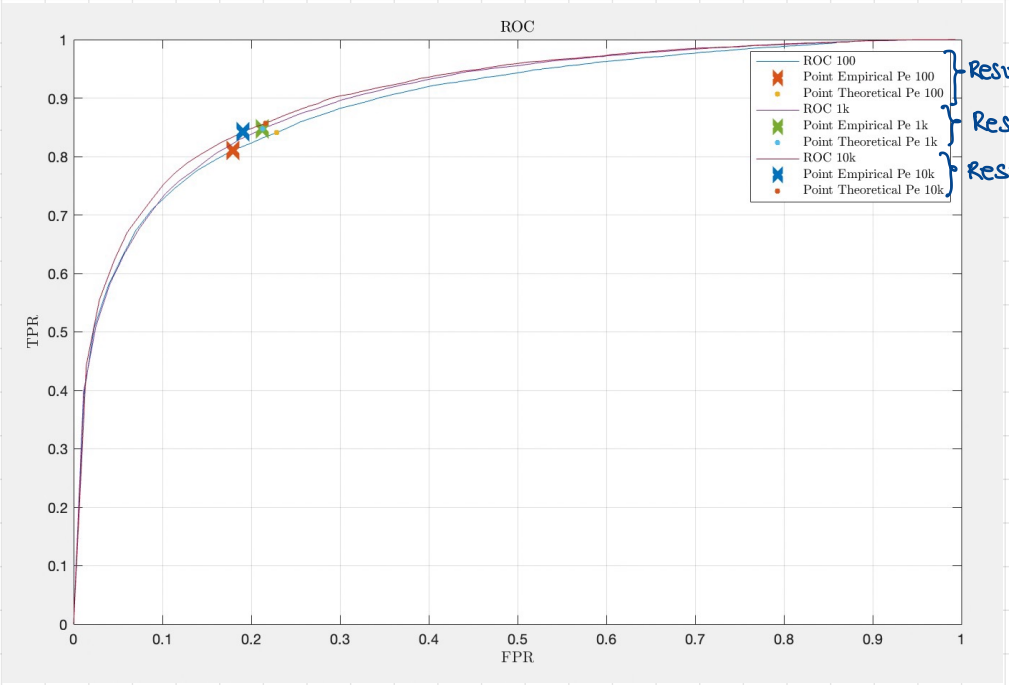
→ a). D_{train}^{10000}

→ b). D_{train}^{1000}

→ c). D_{train}^{100}

After computing the class priors (see explanation above, highlighted in this color), and with the estimate parameters of the Gaussian model (for class 1) and GMM model (for class 0), we can implement the classifier and train it with the 3 training dataset. Then, we perform evaluation with the validation dataset of 20k samples. The results of probability of error and ROC curve when training the classifier with the 3 datasets (always evaluated with the validation dataset) are shown in the following figures.

The classifier is implemented as in Question 1.1. and applied to the validation dataset.



Results obtained with D_{train}^{100}

Results obtained with D_{train}^{1K}

Results obtained with D_{train}^{10K}

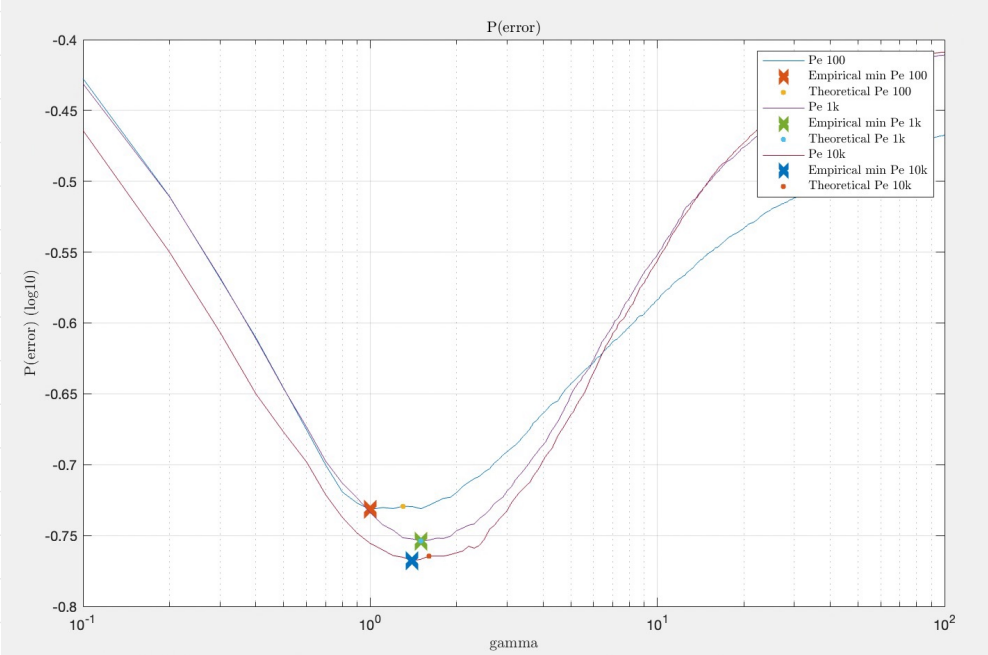
We can see how when the number of training samples decreases, the performance observed via the ROC curve becomes worse and the AUC decreases.

(Empirical values)

N training samples	Threshold	min Pe	Prior Class 0	Prior Class 1
100	1,001	0,1855	0,55	0,45
1000	1,5015	0,1762	0,59	0,4100
10000	1,4014	0,1706	0,6022	0,3978

To estimate the class priors we just check the labels of the training data and calculate proportions.

It is interesting to highlight that for this estimator, the estimate variance is proportional to the variance of the data, so when working with models that have a high variance, this estimator should not be used. This is because, for example, for class 1, we are computing the sample mean to estimate the mean vector, so the more samples in the dataset and with less variance, the better performance we will obtain. Also, when estimating the covariance matrix, we see a performance improvement for a higher number of training samples.



We can see how the probability of error becomes higher when the number of training dataset samples is decreased. This means, for the dataset with lowest number of samples (100) we see a higher probability of error (blue curve) than for the other two datasets.

Question 1- Part 3 (Logistic - Linear)

$x \equiv$ input sample vector

$w \equiv$ model parameter vector

→ **Logistic-Linear-Function:** $h(\underline{x}, \underline{w}) = \overbrace{\frac{1}{1 + e^{-\underline{w}^T \underline{z}(\underline{x})}}}^{\text{Sigmoid function}}$, where $\underline{z}(\underline{x}) = [1, \underline{x}^T]^T$

$\underline{w}^T \underline{z}(\underline{x}) = [w_1 \; w_2 \; w_3] \cdot \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} = w_1 + w_2 \cdot x_1 + w_3 x_2$

(two-dimensional data)

→ $\left.\begin{aligned} p(L_i = 1 \mid \underline{x}_K) &= h(\underline{x}_K, \underline{w}) \\ p(L_i = 0 \mid \underline{x}_K) &= 1 - h(\underline{x}_K, \underline{w}) \end{aligned}\right\} p(L = L_i \mid \underline{x}_K) = h(\underline{x}_K, \underline{w})^{L_i} (1 - h(\underline{x}_K, \underline{w}))^{1-L_i}$

$L(\underline{w}) = p(L \mid \underline{X}) = \prod_{K=1}^{n} h(\underline{x}_K, \underline{w})^{L_i} (1 - h(\underline{x}_K, \underline{w}))^{1-L_i}$ → LIKELIHOOD

($\underline{X}$ ↑ TRAINING DATASET)

$\mathcal{L}(\underline{w}) = \sum_{K=1}^{n} \left(L_i \cdot \ln[h(\underline{x}_K, \underline{w})] + (1 - L_i) \ln[1 - h(\underline{x}_K, \underline{w})] \right)$ → LOG-LIKELIHOOD

$\hat{\underline{w}}_{MLE} = \underset{\underline{w}}{\operatorname{argmax}} \, \mathcal{L}(\underline{w}) \Rightarrow \frac{\partial \mathcal{L}(\underline{w})}{\partial \underline{w}} = 0$

$$\frac{\partial \mathcal{L}(\underline{w})}{\partial \underline{w}} = \sum_{K=1}^{n} \left[\frac{L_i}{h(\underline{x}_K, \underline{w})} \cdot \frac{\partial h(\underline{x}_K, \underline{w})}{\partial \underline{w}} + \frac{(1 - L_i)}{1 - h(\underline{x}_K, \underline{w})} \cdot (-1) \cdot \frac{\partial h(\underline{x}_K, \underline{w})}{\partial \underline{w}} \right] = \sum_{K=1}^{n} \frac{\partial h(\underline{x}_K, \underline{w})}{\partial \underline{w}} \left[\frac{L_i}{h(\underline{x}_K, \underline{w})} - \frac{(1 - L_i)}{1 - h(\underline{x}_K, \underline{w})} \right] = (*)$$

$$\frac{\partial h(\underline{x}_K, \underline{w})}{\partial \underline{w}} = \frac{\partial h(\underline{x}_K, \underline{w})}{\partial \underline{w}^T \underline{x}_K} \cdot \underbrace{\frac{\partial \underline{w}^T \underline{x}_K}{\partial \underline{w}}}_{\underline{x}_K} = h(\underline{x}_K, \underline{w}) (1 - h(\underline{x}_K, \underline{w})) \underline{x}_K$$

$$(*) = \sum_{K=1}^{n} h(\underline{x}_K, \underline{w}) (1 - h(\underline{x}_K, \underline{w})) \underline{x}_K \left[\frac{L_i (1 - h(\underline{x}_K, \underline{w})) - (1 - L_i) h(\underline{x}_K, \underline{w})}{h(\underline{x}_K, \underline{w})(1 - h(\underline{x}_K, \underline{w}))} \right] =$$

$$= \sum_{K=1}^{n} \underline{x}_K \left(L_i - L_i \cdot h(\underline{x}_K, \underline{w}) + L_i h(\underline{x}_K, \underline{w}) - h(\underline{x}_K, \underline{w}) \right) = \boxed{\sum_{K=1}^{n} (L_i - h(\underline{x}_K, \underline{w})) \underline{x}_K = \frac{\partial \mathcal{L}(\underline{w})}{\partial \underline{w}}}$$

In the exercise it is asked to compute this expression by $\hat{w}_{ML} = \underset{\underline{w}}{\operatorname{argmin}} \; -\frac{1}{N} \sum_{K=1}^{n} [L_i \ln h(\underline{x}_K, \underline{w}) + (1 - L_i) \cdot \ln(1 - h(\underline{x}_K, \underline{w}))]$, but I am maximizing the log-likelihood instead.

The resulting equation could be used for Gradient Ascent (GA) to compute the values of $\underline{w}$ until we converge:

$$\underline{w}^t = \underline{w}^{t-1} + \alpha \frac{\partial \mathcal{L}(\underline{w})}{\partial \underline{w}}$$

(α → SCALING PARAMETER (step size))

→ Logistic-Quadratic-Function: $h(\underline{x}, \underline{w}) = \frac{1}{1 + e^{-\underline{w}^T \underline{z}(\underline{x})}}$, where $\underline{z}(\underline{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix}$

$$\underline{w}^T \underline{z}(\underline{x}) = [w_1\ w_2\ w_3\ w_4\ w_5\ w_6] \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix} = w_1 + w_2 x_1 + w_3 x_2 + w_4 x_1^2 + w_5 x_1 x_2 + w_6 x_2^2$$

They propose to approximate the class label posterior function given each sample.

→ Same math as in previous exercise

parameters, specify the optimization problem as minimization of the negative-log-likelihood of the training dataset, and use your favorite numerical optimization approach, such as gradient descent or Matlab's *fminsearch* or Python's *minimize*. Use the trained class-label-posterior approximations to classify validation samples to approximate the minimum-P(error) classification rule; estimate the probability of error that these three classifiers attain using counts of decisions on the validation set. Optional: As supplementary visualization, generate plots of the decision boundaries of these trained classifiers superimposed on their respective training datasets and the validation dataset. (b) Repeat the process described in Part (3a) using a logistic-quadratic-function-based approximation of class label posterior functions given a sample.

For this question, we want to estimate the probability of error that the three classifiers, for each type of logistic function, provide by using the validation dataset.

N training samples	Logistic-Linear	Logistic-quadratic
100	0,6123	0,1756
1000	0,5856	0,1722
10000	0,5459	0,1705

I have trained the model with both the logistic linear and logistic quadratic function, with the three different training datasets, and always validating the model with the validation dataset, which contains 20000 samples. I have used the Matlab function fminsearch to find the values of the weights (parameter vector to estimate) that minimize the resulting cost function (which has been mathematically derived in the previous page). The initializatioin of the weights is set to zero value (as recommended by the professor).

The results presented in the table suggest that the logisitc-quadratic model performs better than the logistic-linear model, which makes sense, as the quadratic model uses a polynomial of more degree in the sigmoid function, giving more freedom when building the boundary. Therefore, by increasing the number of training samples we can approach the Pe that we observe in the classifier from question 1.2 (but we never completely achieve this performance because of the limitation of having a - yes or yes - quadratic polynomial boundary).

General Conclusion:
We can see how the classifiers in this question perform worse than the classifiers in the previous question. This is because the classifiers given by the linear and quadratic polynomials have a **BOUNDED PERFORMANCE**, which cannot be overcome via the increase of training samples. The decision boundary will always be either linear or quadratic, no matter how many training samples are used, so the performance will never reach the one provided by the classifier operatinc point in questions 1.1 and 1.2 (theoretically optimal classification rule). The classification boundary that gives best performance is actually complex (we see this in question 1.1), so it needs more degrees for the polynomial.

Question 2 (30%)

A vehicle at true position $[x_T, y_T]^{\mathrm{T}}$ in 2-dimensional space is to be localized using distance (range) measurements to K reference (landmark) coordinates $\{[x_1, y_1]^{\mathrm{T}}, \ldots, [x_i, y_i]^{\mathrm{T}}, \ldots, [x_K, y_K]^{\mathrm{T}}\}$. These range measurements are $r_i = d_{Ti} + n_i$ for $i \in \{1, \ldots, K\}$, where $d_{Ti} = \|[x_T, y_T]^{\mathrm{T}} - [x_i, y_i]^{\mathrm{T}}\|$ is the true distance between the vehicle and the i^{th} reference point, and n_i is a zero mean Gaussian distributed measurement noise with known variance σ_i^2. The noise in each measurement is independent from the others.

Assume that we have the following prior knowledge regarding the position of the vehicle:

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

value of σ_x, σ_y relates to the prior confidence

where $[x, y]^{\mathrm{T}}$ indicates a candidate position under consideration.

Express the optimization problem that needs to be solved to determine the MAP estimate of the vehicle position. Simplify the objective function so that the exponentials and additive/multiplicative terms that do not impact the determination of the MAP estimate $[x_{MAP}, y_{MAP}]^{\mathrm{T}}$ are removed appropriately from the objective function for computational savings when evaluating the objective.

Implement the following as computer code: Set the true vehicle location to be inside the circle with unit radious centered at the origin. For each $K \in \{1, 2, 3, 4\}$ repeat the following.

Place evenly spaced K landmarks on a circle with unit radius centered at the origin. Set measurement noise standard deviation to 0.3 for all range measurements. Generate K range measurements according to the model specified above (if a range measurement turns out to be negative, reject it and resample; all range measurements need to be nonnegative).

Plot the equilevel contours of the MAP estimation objective for the range of horizontal and vertical coordinates from -2 to 2; superimpose the true location of the vehicle on these equilevel contours (e.g. use a $+$ mark), as well as the landmark locations (e.g. use a o mark for each one).

Provide plots of the MAP objective function contours for each value of K. When preparing your final contour plots for different K values, make sure to plot contours at the same function value across each of the different contour plots for easy visual comparison of the MAP objective landscapes. *Suggestion:* For values of σ_x and σ_y, you could use values around 0.25 and perhaps make them equal to each other. Note that your choice of these indicates how confident the prior is about the origin as the location.

Supplement your plots with a brief description of how your code works. Comment on the behavior of the MAP estimate of position (visually assessed from the contour plots; roughly center of the innermost contour) relative to the true position. Does the MAP estimate get closer to the true position as K increases? Doe is get more certain? Explain how your contours justify your conclusions.

Note: The additive Gaussian distributed noise used in this question is actually not appropriate, since it could lead to negatuve measurements, which are not legitimate for a proper distance sensor. However, in this question, we will ignore this issue and proceeding with this noise model for the sake of illustration. In practice, a multiplicative log-normal distributed noise may be more appropriate than an additive normal distributed noise.

Question 2 - Part 1

Question 2 (30%)

A vehicle at true position $[x_T, y_T]^T$ in 2-dimensional space is to be localized using distance (range) measurements to K reference (landmark) coordinates $\{[x_1, y_1]^T, \ldots, [x_i, y_i]^T, \ldots, [x_K, y_K]^T\}$. These range measurements are $r_i = d_{Ti} + n_i$ for $i \in \{1, \ldots, K\}$, where $d_{Ti} = \|[x_T, y_T]^T - [x_i, y_i]^T\|$ is the true distance between the vehicle and the i^{th} reference point, and n_i is a zero mean Gaussian distributed measurement noise with known variance σ_i^2. The noise in each measurement is independent from the others.

Assume that we have the following prior knowledge regarding the position of the vehicle:

$$p(\underline{s}_T) \leftarrow p\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = (2\pi\sigma_x\sigma_y)^{-1} e^{-\frac{1}{2}\begin{bmatrix} x & y \end{bmatrix}\underbrace{\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1}}_{\underline{\underline{C}}}\begin{bmatrix} x \\ y \end{bmatrix}} \quad (1)$$

where $[x, y]^T$ indicates a candidate position under consideration.

Express the optimization problem that needs to be solved to determine the MAP estimate of the vehicle position. Simplify the objective function so that the exponentials and additive/multiplicative terms that do not impact the determination of the MAP estimate $[x_{MAP}, y_{MAP}]^T$ are removed appropriately from the objective function for computational savings when evaluating the objective.

$\underline{s}_T \equiv$ true position

$\underline{s} = [x \; y]^T$

$\underline{s}_T = [x_T \; y_T]^T$

pos. landmark $i \rightarrow \underline{s}_{Li} = [x_i \; y_i]^T$

$n_i \sim \mathcal{N}(0, \sigma_i^2)$ (noise)

$r_i \sim \mathcal{N}(d_{Ti}, \sigma_i^2)$ (measured position)

$\underline{r} = (r_1, \ldots r_K)$

$\underline{d}_T = [d_{T_1}, \ldots, d_{T_2}]$

With the Maximum a Posteriori (MAP) estimator, we find the value of the parameter vector to estimate that maximizes the posterior probability. Therefore, we first need to compute the posterior probability. With the Bayes rule, we see that the posterior can be expressed as the product of the likelihood by the prior divided by the evidence. We first need to understand how to build the likelihood with the information given in this exercise as:

$p(x|\theta)$ → parameter to estimate (position $\underline{s}_T$) → likelihood is $p(\underline{r}|\underline{s}_T)$

↓ measurements $\underline{r}$

Likelihood for one landmark would be $p(r_i) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{1}{2}\left(\frac{r_i - d_{Ti}}{\sigma_i}\right)^2} = p(r_i|\underline{s}_T)$

Likelihood for $i = 1, \ldots, K$ landmarks (i.i.d) would be $p(\underline{r}) = \prod_{i=1}^{K} \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{1}{2}\left(\frac{r_i - d_{Ti}}{\sigma_i}\right)^2\right)$

The likelihood that considers the ranges (which are measurements, observations of this problem) between the vehicle and all the landmarks corresponds to p(**r**), being **r** a vector containing all these ranges. Consequently, we have a multivariate distribution (we see the product symbol, which will become a sum when computing the log-likelihood). Then, we can obtain the MAP estimator for s_t (true position of the vehicle) as follows.

The objective function that we find is alpha(s_t). As asked in the exercise, we manipulate this expression (i.e. the objective function) to remove all the terms that will not have an impact when maximizing the posterior (this means, terms that do not actually depend on s_t). In the next page, we calculate the logarithm of the objective function, getting terms from the log-likelihood and from the log-prior. As the observed range r_i depends on the true position of the vehicle, if an arbitrary candidate position is closer to the true position, the likelihood will be higher. Also, it is interesting to highlight that the prior is suggesting that the vehicle is located at the center of the origin (mean [0, 0]) with certain uncertainty. This is why we are expecting the contours to be around [0, 0] when the MAP estimator relies more on the prior (for example when the number of landmarks is low).

$$\hat{\underline{s}}_{T\,MAP} = \underset{\underline{s}_T}{\operatorname{argmax}}\, p(\underline{s}_T|\underline{r}) \overset{\text{BAYES RULE}}{=} \underset{\underline{s}_T}{\operatorname{argmax}}\, \frac{p(\underline{r}|\underline{s}_T)\,p(\underline{s}_T)}{p(\underline{r})} = \underset{\underline{s}_T}{\operatorname{argmax}}\, p(\underline{r}|\underline{s}_T)\,p(\underline{s}_T) =$$

$$= \underset{\underline{s}_T}{\operatorname{argmax}} \underbrace{\prod_{i=1}^{K} \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{1}{2}\left(\frac{r_i - d_{Ti}}{\sigma_i}\right)^2\right) \cdot \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}\underline{s}_T^T \underline{\underline{C}}^{-1} \underline{s}_T}}_{\alpha(\underline{s}_T) \equiv \text{objective function}}$$

Evidence $p(\underline{r})$ does not depend on the parameter to estimate.

Also, I would like to highlight that although we are maximizing the posterior, we end up trying to find a minimum because we multiply by (-1) the expression. This is important because when we see the contours we will see that values become smaller for rings of lower radius. The function alpha_prime(s_t) contains the objective function after manipulation of the terms:

$$\rightarrow \alpha(\underline{s}_T) = \prod_{i=1}^{K} \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{1}{2}\left(\frac{r_i - d_{Ti}}{\sigma_i}\right)^2\right) \cdot \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}\underline{s}_T^T \underline{\underline{C}}^{-1} \underline{s}_T}$$

$$\rightarrow \ln(\alpha(\underline{s}_T)) = \sum_{i=1}^{K}\left[-\ln(\sqrt{2\pi}\sigma_i) - \frac{1}{2}\left(\frac{r_i - d_{Ti}}{\sigma_i}\right)^2\right] - \ln(2\pi\sigma_x\sigma_y) - \frac{1}{2}\underline{s}_T^T \underline{\underline{C}}^{-1} \underline{s}_T =$$

this terms do not impact the maximization, so we can avoid them

$$= \sum_{i=1}^{K} -\frac{1}{2}\left(\frac{r_i - d_{Ti}}{\sigma_i}\right)^2 - \frac{1}{2}\underline{s}_T^T \underline{\underline{C}}^{-1} \underline{s}_T$$

$$\rightarrow \hat{\underline{s}}_{T_{MAP}} = \underset{\underline{s}_T}{\text{argmax}} \ln(\alpha(\underline{s}_T)) = \underset{\underline{s}_T}{\text{argmin}} \left[\sum_{i=1}^{K} \frac{1}{2}\left(\frac{r_i - d_{Ti}}{\sigma_i}\right)^2 + \frac{1}{2}\underline{s}_T^T \underline{\underline{C}}^{-1} \underline{s}_T\right] =$$

$$= \underset{\underline{s}_T}{\text{argmin}} \left[\sum_{i=1}^{K} \left(\frac{r_i - \|\underline{s}_T - \underline{s}_{Li}\|}{\sigma_i}\right)^2 + \underline{s}_T^T \underline{\underline{C}}^{-1} \underline{s}_T\right]$$

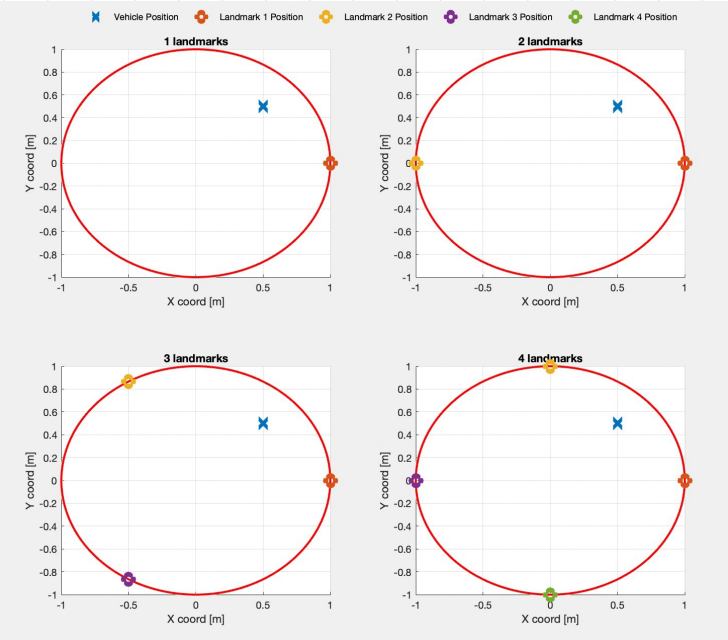
$\alpha'(\underline{s}_T)$ = objective function after removing the unnecessary terms and also applying logarithm

Question 2 - Part 2 (Implementation with MATLAB)

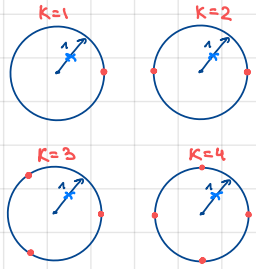
Implement the following as computer code: Set the true vehicle location to be inside the circle with unit radious centered at the origin. For each $K \in \{1,2,3,4\}$ repeat the following.

Place evenly spaced K landmarks on a circle with unit radius centered at the origin. Set measurement noise standard deviation to 0.3 for all range measurements. Generate K range measurements according to the model specified above (if a range measurement turns out to be negative, reject it and resample; all range measurements need to be nonnegative).

So, first of all I have used the values recommended in the problem statement, as seen in the following screenshot. I have decided the position of the vehicle to be at x=0.5, y=0.5. Also, sigma_i = 0.3, sigma_x=sigma_y=0.25. I am assuming that all these values are given in meters (see plot labels). The scenario I am generating for this first case can be seen here:



Proposed landmark and vehicle positions:




```
%% Generate scenario

% True vehicle position
s_t = [0.5 0.5]'; % we define position vectors as column vectors (like in the exercise)

% Landmark coordinates (hardcoded)
landmark.pos{1} = [1 0]';
landmark.pos{2} = [1 -1; 0 0];
landmark.pos{3} = [1 -0.5 -0.5; 0 0.866 -0.866];
landmark.pos{4}  = [1 0 -1 0; 0 1 0 -1];
landmark.num = 4; % number of landmarks in the scenario

% Standard deviation values
sigma_i = 0.3;
sigma_x = 0.25; % sigma_x and sigma_y are related to the prior confidence
sigma_y = 0.25;
C = [sigma_x^2 0; 0 sigma_y^2];
```

reject it and resample; all range measurements need to be nonnegative).

Plot the equilevel contours of the MAP estimation objective for the range of horizontal and vertical coordinates from -2 to 2; superimpose the true location of the vehicle on these equilevel contours (e.g. use a $+$ mark), as well as the landmark locations (e.g. use a o mark for each one).

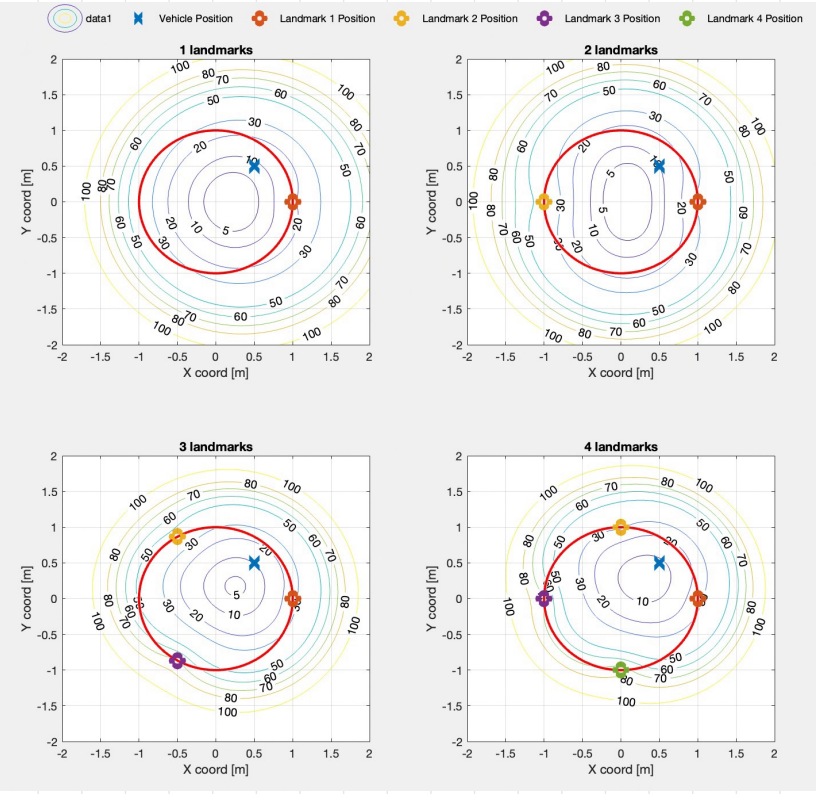
Provide plots of the MAP objective function contours for each value of K. When preparing your final contour plots for different K values, make sure to plot contours at the same function value across each of the different contour plots for easy visual comparison of the MAP objective landscapes. *Suggestion:* For values of σ_x and σ_y, you could use values around 0.25 and perhaps make them equal to each other. Note that your choice of these indicates how confident the prior is about the origin as the location.

Supplement your plots with a brief description of how your code works. Comment on the behavior of the MAP estimate of position (visually assessed from the contour plots; roughly center of the innermost contour) relative to the true position. Does the MAP estimate get closer to the true position as K increases? Doe is get more certain? Explain how your contours justify your conclusions.

My code is first generating the scenario that has just been shown, locating the vehicle and then the landmarks equally spaced on the unit circle centered at the origin. I am performing 4 experiments, each one with a different number of landmarks. For each experiment, and each landmark, the range between the vehicle (true position, s_t) and the landmark position is computed and then a gaussian noise is added up, making sure that the computed range (which we assume to be measured) follows the nonnegative condition. Then, a mesh is built, with a number of points that can be changed in the code (I decide this number of points to be 1e3). This mesh covers the range from -2 to 2 both in the x and y coordinates (assumed to be in meters). Then, the manipulated (without unnecessary terms) objective function alpha_prime(s_t) is evaluated at each of the points of the created mesh. The result of this is computed with the contour MATLAB function, giving the value of the maximization of the posterior distribution at each value of the mesh in contours (equilevel contours). As it has been previously explained, although we are maximizing the posterior, we end up minimizing the manipulated objective function because of multiplying the expression by -1, so we can see how the values of the contours become smaller (we are trying to find a minimum).

When we increase the number of landmarks (from 1 to 4) we see an improvement in the MAP estimator, as the center of the innermost contour becomes closer to the true position, marked with a blue cross marker. Also, with a higher value of k landmarks, we see that the estimation is not converging towards [0, 0], which is what happens with low number of landmarks (k=1,2) because the prior brings the contours towards the center of the circle (the prior is biasing the estimator). The level of bias that the prior gives to the estimator depends on the terms sigma_x and sigma_y, while sigma_i has an effect of how uncertain the likelihood is. We can actually play with these parameters (see next page).

Results for s_t = [0.5, 0.5], sigma_x = sigma_y = 0.25 and sigma_i = 0.3:



1.5592

1.1557

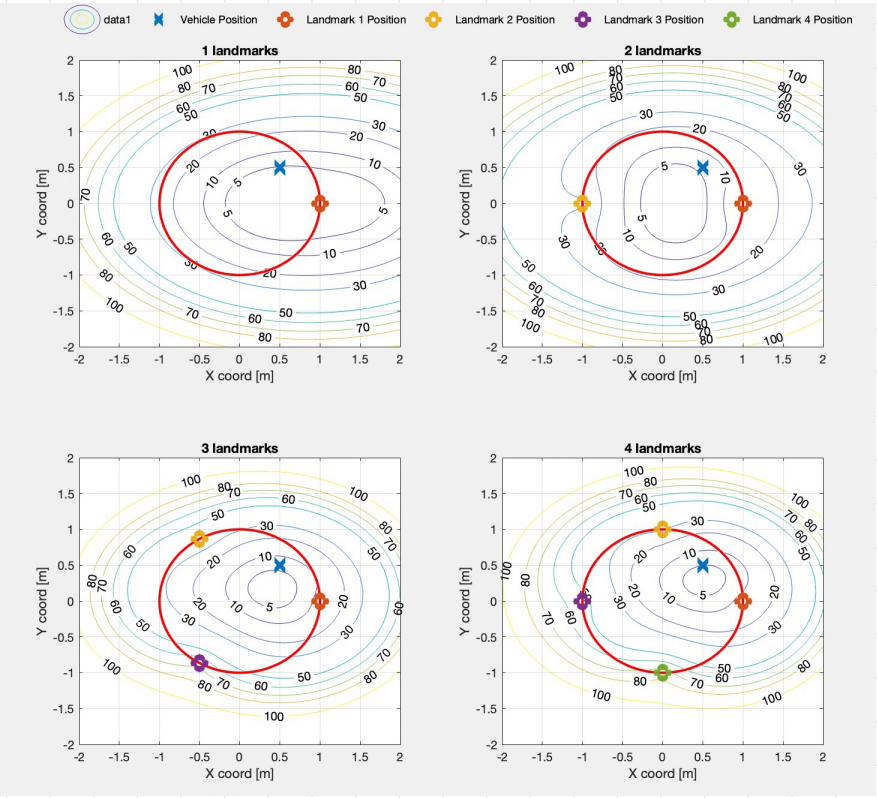
4.3444

5.7863

Minimum value of the MAP contours for $K = 1, \dots, 4$.

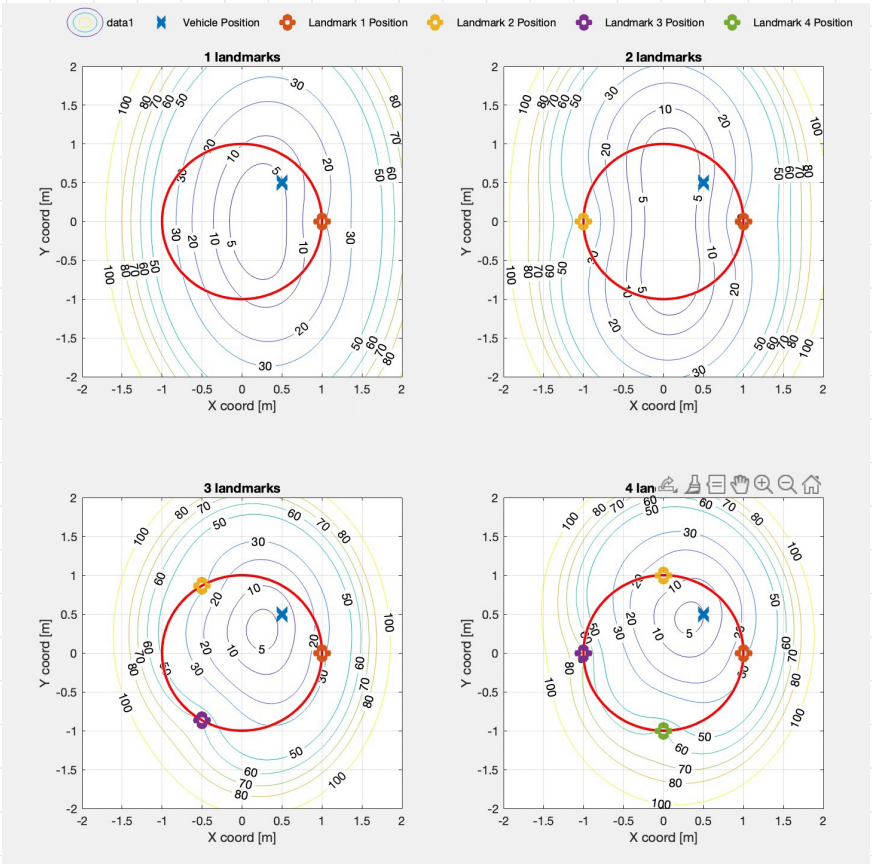
In order to be able to compare contours for different number of landmarks, I am plotting the contour values. We can see how the innermost contour converges to the true position for an increasing value of landmarks. However, it is interesting to see that the minimum value of the MAP contour becomes bigger (this means, a less minimum point, because we are minimizing the expression) when the number of landmarks is increased. See for example that for k=3 we have a contour with value 5, while for k=4 the minimum MAP value computed is higher than 5.

Results when increasing sigma_x to 0.9:



In this page I am trying to show how by changing the terms sigma_x and sigma_y we can see an effect on the uncertainty that the prior can bring to the estimator. So, when sigma_x is increased, we see that the contours become wider (the variance expands along the horizontal axis), while when sigma_y is increased, the contours expand along the y axis (vertically).

Results when increasing sigma_y to 0.9 (sigma_x = 0.25 as originally):



Question 3 (20%)

Problem 2.13 from Duda-Hart-Stork textbook:

Section 2.4

13. In many pattern classification problems one has the option either to assign the pattern to one of c classes, or to *reject* it as being unrecognizable. If the cost for rejects is not too high, rejection may be a desirable action. Let

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & i = j \qquad i, j = 1, \ldots, c \\ \lambda_r & i = c + 1 \\ \lambda_s & \text{otherwise,} \end{cases}$$

where λ_r is the loss incurred for choosing the $(c + 1)$th action, rejection, and λ_s is the loss incurred for making any substitution error. Show that the minimum risk is obtained if we decide ω_i if $P(\omega_i|\mathbf{x}) \geq P(\omega_j|\mathbf{x})$ for all j and if $P(\omega_i|\mathbf{x}) \geq 1 - \lambda_r/\lambda_s$, and reject otherwise. What happens if $\lambda_r = 0$? What happens if $\lambda_r > \lambda_s$?

Question 3

→ We compute the risk of taking action α_i as:

$$R(\alpha_i|\underline{x}) = \sum_{j=1}^{c} \lambda(\alpha_i|w_j)\, p(w_j|\underline{x}) = \sum_{j\neq i}^{c} \lambda_s\, p(w_j|\underline{x})$$

$$= \lambda_s \sum_{j\neq i}^{c} p(w_j|\underline{x}) = \lambda_s (1 - p(w_i|\underline{x}))$$

→ The risk of taking decision α_{c+1}:

$$R(\alpha_{c+1}|\underline{x}) = \sum_{j=1}^{c} \lambda(\alpha_{c+1}|w_j)\, p(w_j|\underline{x}) = \sum_{j=1}^{c} \lambda_r\, p(w_j|\underline{x}) = \lambda_r \sum_{j=1}^{c} p(w_j|\underline{x}) = \lambda_r$$

In order to decide whether we take action α_i or not, we check if its risk is smaller than the risk of taking $\alpha_{j|_{j\neq i}}$ (i.e. any other possible action).

We check this with the following ratio:

$$R(\alpha_i|\underline{x}) \overset{?}{\leq} R(\alpha_j|\underline{x}) \quad \forall j = 1, \dots, c \quad \text{and } j \neq i$$

With the previous calculations we have that:

$$\lambda_s(1 - p(w_i|\underline{x})) \leq \lambda_s (1 - p(w_j|\underline{x})) \Rightarrow \boxed{p(w_i|\underline{x}) \geq p(w_j|\underline{x}) \quad \forall j = 1, \dots, c \text{ and } j \neq i}$$

→ Important Finding 1

We use the second calculation to state that:

$$R(\alpha_i|\underline{x}) \leq R(\alpha_{c+1}|\underline{x})$$

$$\lambda_s(1 - p(w_i|\underline{x})) \leq \lambda_r \Rightarrow \boxed{p(w_i|\underline{x}) \geq 1 - \frac{\lambda_r}{\lambda_s}}$$

→ Important Finding 2, where we have considered the rejection risk.

Conclusions/insights:

As mentioned before, the risk of action alpha_i must be lower than the risk of rejection (this is a necessary condition). When applying this condition, we obtain result (2).

- Given result (2), if lambda_r is 0, the decision will always be rejected for all the values of x.
- The higher lambda_r is with respect to lambda_s (see the lambda_r/lambda_s ratio in result 2), we have less chances of rejecting the decision.
- If lambda_r = lambda_s, the class that will be selected is the class with the maximum posterior probability, given result (1).