Assignment – 2
EECE 5644
Prayag Sridhar

Q1 Part 1: Bayes classification using true PDF

As per the question it is required to initially identify the theoretically ideal classifier using the provided probability density functions (pdf) for the two-dimensional random vector X. By utilizing the known class priors and mixture parameters, we mathematically derive and execute the Bayes classifier to minimize the probability of error. The classifier is subsequently assessed using a validation dataset (D10K) with 10,000 samples sourced from the actual class-conditional distributions.

The probability density functions conditional on the classes for the two groups are established as:
p(x|L=0) = 0.5 * N(x; m01, C) + 0.5 * N(x; m02, C)
p(x|L=1) = 0.5 * N(x; m11, C) + 0.5 * N(x; m12, C)

where **m01 = [-0.9, -1.1]^T, m02 = [0.75, 0.8]^T, m11 = [-1.1, 0.9]^T, m12 = [0.9, -0.75]^T, C = diag(0.75, 1.25)** and the prior probabilities are P(L=0) = 0.6, P(L=1) = 0.4

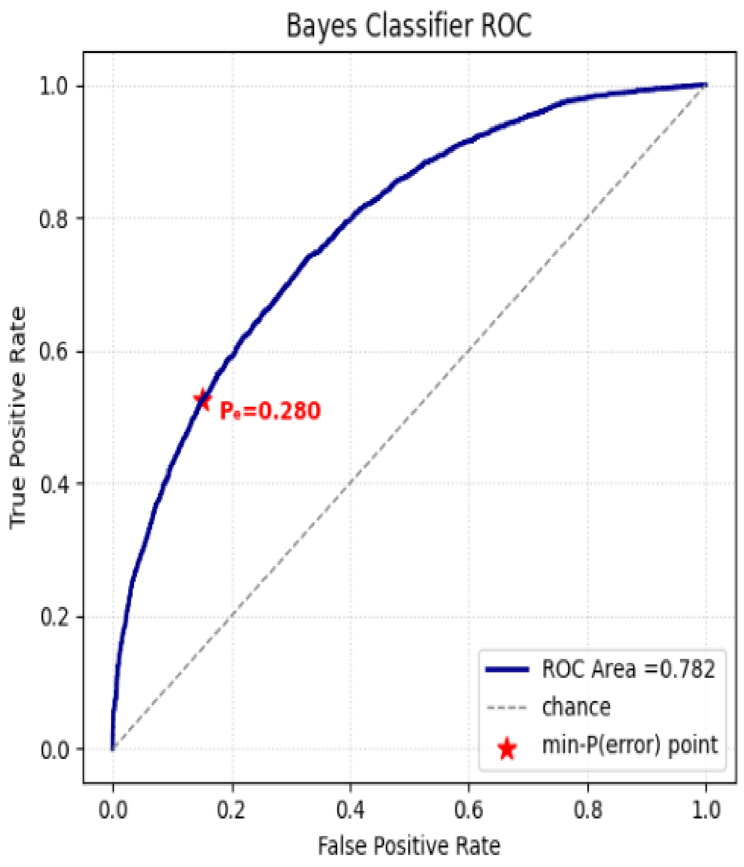
The Bayes decision rule is applied through the log-likelihood ratio test:
log[p(x|L=1) * P(L=1)] - log[p(x|L=0) * P(L=0)] > 0 => Choose L=1
log[p(x|L=1) * P(L=1)] - log[p(x|L=0) * P(L=0)] < 0 => Choose L=0

This guideline reduces the anticipated 0–1 loss and offers the minimum attainable likelihood of misclassification for the specified distributions.

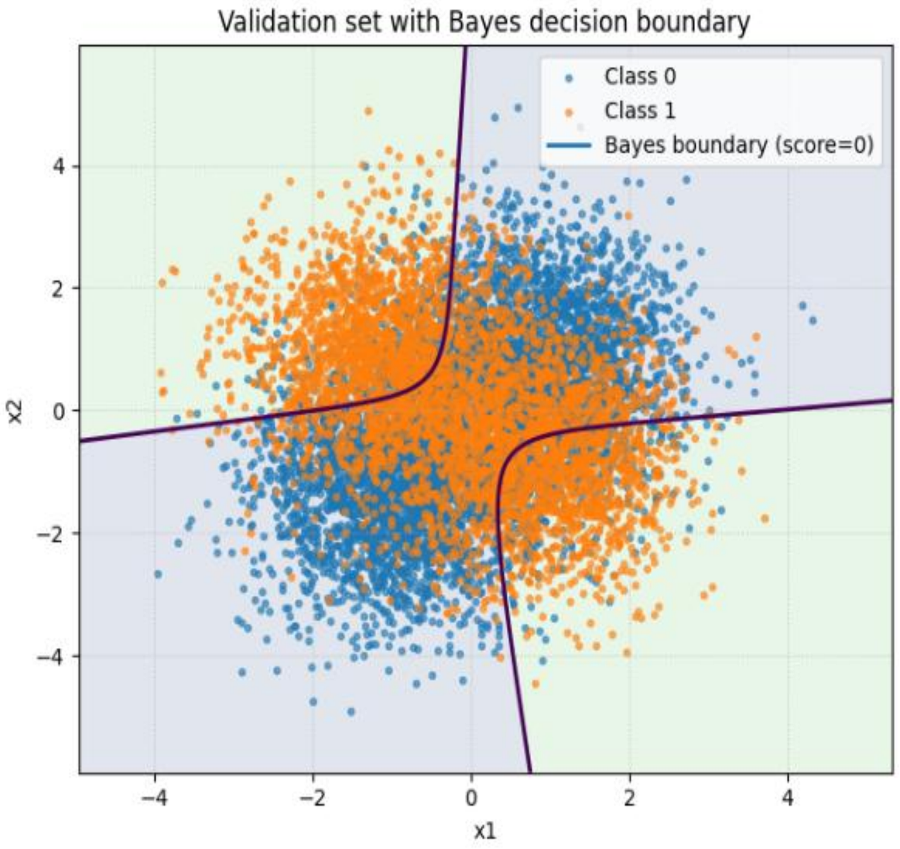
As per the mixture model and class priors, a total of 10,000 validation samples were produced. For every sample, the discriminant score s(x) was calculated, and ROC curve was generated by adjusting the decision threshold. The ROC curve distinctly surpasses the random-guess diagonal, demonstrating efficient separation of the two classes. The curve showed the minimum-error operating point, which was highlighted with a red star. The related empirical minimum probability of error was determined to be around:

P_e (min) = 0.28.

```
Estimated min P(error) on D10K validate: 0.2797
Threshold at min-error: 0.01060
FPR=0.1499, TPR=0.5250:
ROC Area: 0.7817
```



This indicates that the Bayes classifier accurately categorizes approximately 72% of the validation samples. An AUC value near 0.78 indicates a moderate degree of separability between the two classes: their distributions exhibit overlap, yet the optimal classifier outperforms random guessing considerably



The chart above displays the validation data set with the Bayes decision boundary superimposed. Every point denotes a sample from either of the two Gaussian mixture classes — blue signifies Class 0 and orange signifies Class 1. The solid curve represents the boundary where the Bayes discriminant score s(x)=0, delineating the two decision areas.

The boundary appears to be non-linear, curving around the areas where the two mixtures overlap. This shape precisely represents the intricate class distributions and verifies that the Bayes classifier adjusts to the fundamental probability framework instead of presuming a straightforward linear division. The visualization demonstrates why the classifier reaches the lowest theoretical error probability identified in Part 1.

Question 1 (part 2a & 2b): Logistic Regression Classifiers

In this part, we approximate the posterior probability P(L=1|x) using logistic regression models trained by maximum-likelihood estimation (MLE).

We consider two functional forms: logistic-linear model and logistic-quadratic model and evaluate their empirical error on the same validation set $D^{10K}_{\text{validate}}$

(a) Logistic-Linear Posterior Approximation
I approximated the posterior probability $P(L = 1 \mid x)$ using a logistic-linear model:

$h(x; w) = 1 / (1 + \exp(-w^T z(x)))$ where **$z(x) = [1, x_1, x_2]^T$.**

The parameter w is estimated by minimizing the negative log-likelihood (NLL):

$L(w) = -\Sigma_n [y_n \ln h(x_n; w) + (1 - y_n) \ln(1 - h(x_n; w))]$.

The classifier follows the minimum-error rule with equal costs:

$\acute{L}(x) = 1$ if $h(x; w) >= 0.5$, otherwise **$\acute{L}(x) = 0$.**

The model was trained on three datasets D_{50}^{train}, D_{500}^{train}, and D_{5000}^{train}. The corresponding training errors were:

Dataset	**P(error)**
D_{50}	0.480
D_{500}	0.412
D_{5000}	0.382

With the rise in training samples, the error rate slowly diminished, suggesting a more trustworthy parameter estimation. Nonetheless, despite having 5000 samples, the linear classifier was unable to completely represent the curved boundary separating the two Gaussian-mixture classes.
The decision boundary often slices through areas of high overlap, causing incorrect classifications. The visual representations distinctly illustrate a single straight line, indicating that this model fails to capture the non-linear data pattern and resulting in increased error rates.

(b) Logistic-Quadratic Posterior Approximation
To model curved boundaries, the feature vector was extended with quadratic terms:

$z(x) = [1, x_1, x_2, x_1^2, x_1x_2, x_2^2]^T$.

The identical training process is utilized with the previously mentioned negative log-likelihood objective. This model can depict curved decision boundaries that align more closely with the actual class distributions.

The errors from the training were:

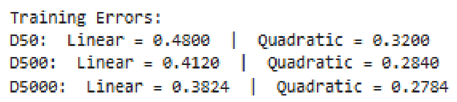
Dataset	**P(error)**
D_{50}	0.320

D_{500}	0.284
D_{5000}	0.278

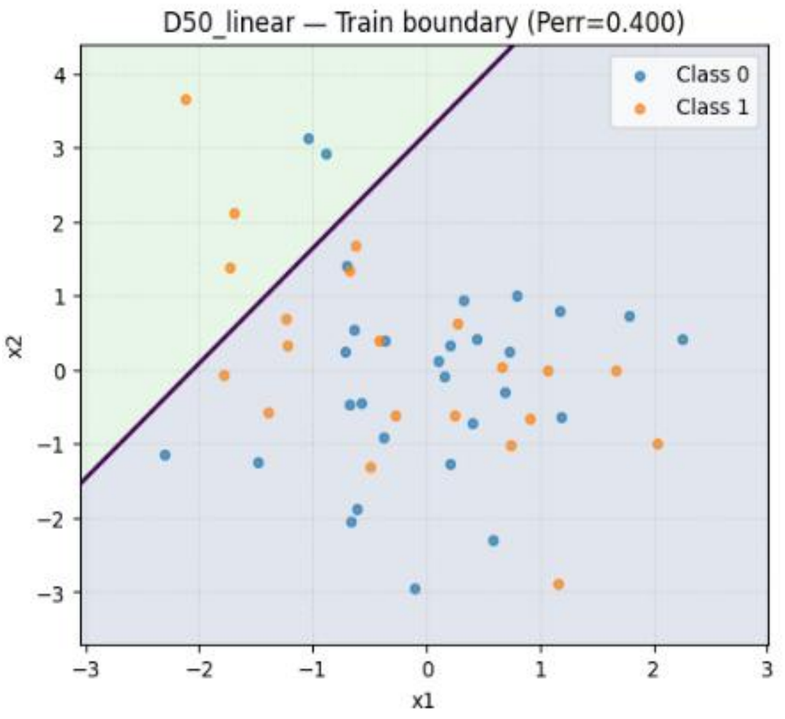
The quadratic classifier consistently yielded reduced error rates compared to the linear one across all dataset sizes. Its boundary curves around the data groupings and adapts to the underlying Gaussian mixture structure. With the rise in sample size, the decision surface turned smoother and more consistent.

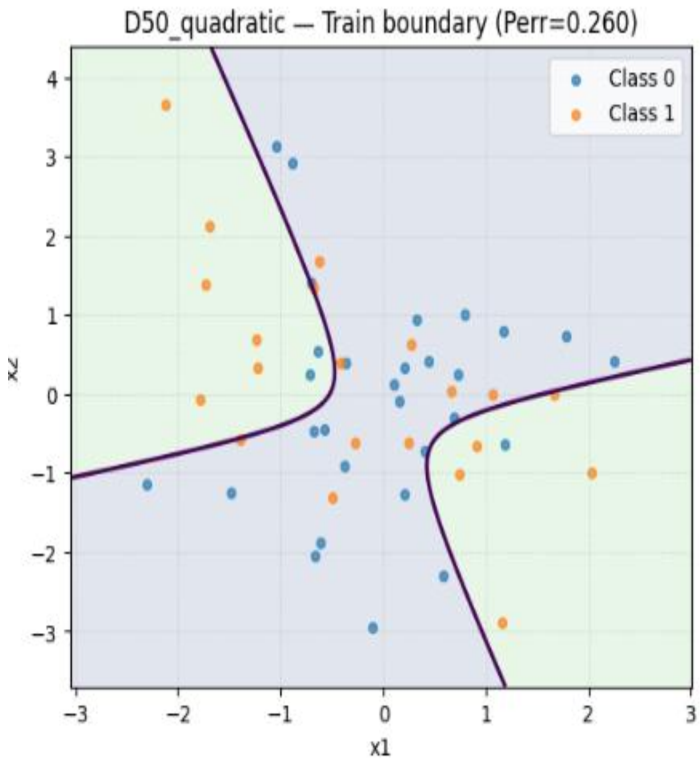
In the largest dataset (D_{5000}), the error rate of approximately 0.278 is nearly the same as the Bayes minimum error $P_e(\min) = 0.28$ achieved in Part 1.

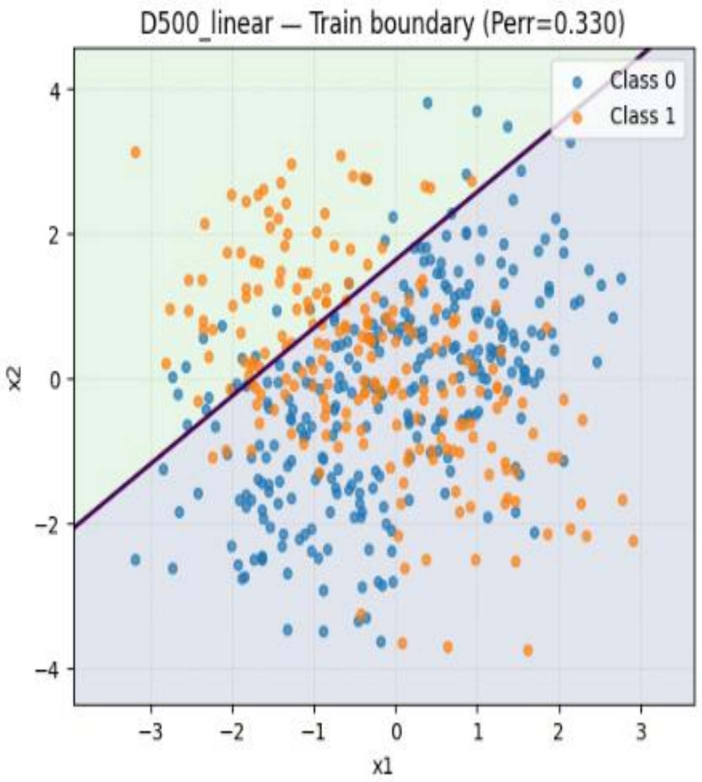
This indicates that the quadratic logistic model can closely match the theoretically ideal classifier when sufficient data is present. The logistic-quadratic classifier adjusts to curved class limits and achieves results significantly nearer to the Bayes-optimal outcome.

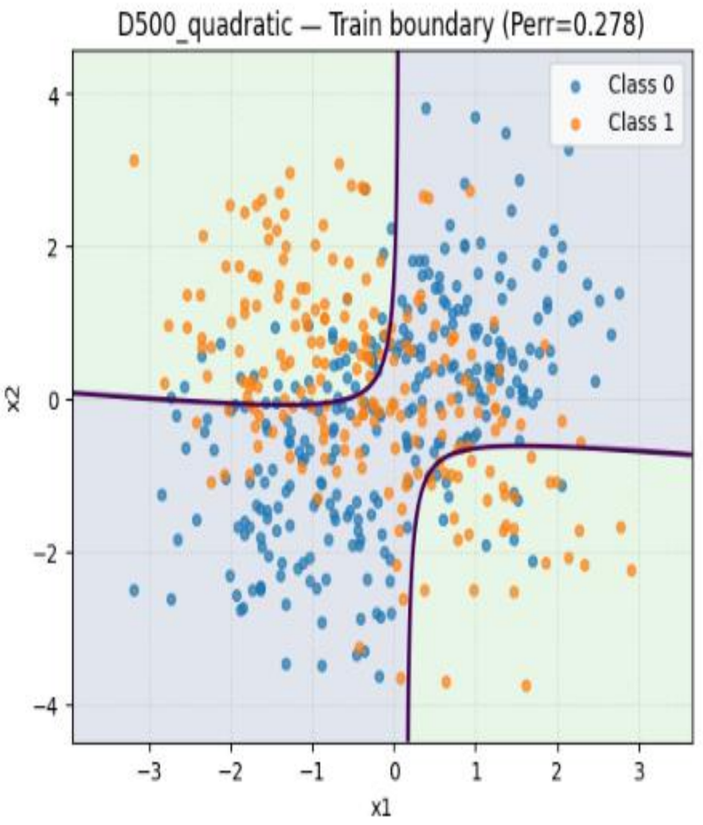


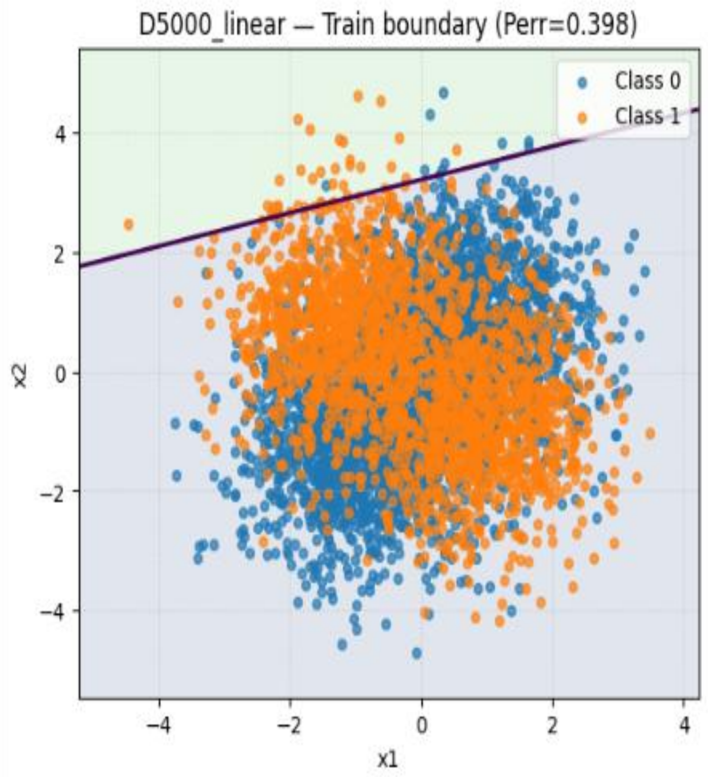
```
Training Errors:
D50:  Linear = 0.4800  |  Quadratic = 0.3200
D500:  Linear = 0.4120  |  Quadratic = 0.2840
D5000:  Linear = 0.3824  |  Quadratic = 0.2784
```

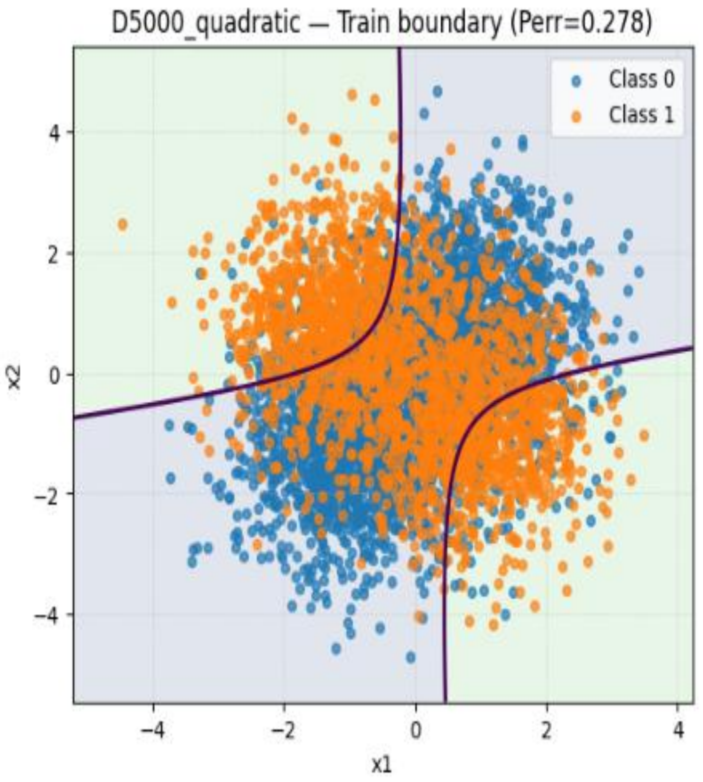



D50_quadratic — Train boundary (Perr=0.260)
Class 0
Class 1
x1
x2


D500_linear — Train boundary (Perr=0.330)
Class 0
Class 1
x1
x2


D500_quadratic — Train boundary (Perr=0.278)
Class 0
Class 1
x1
x2
−3
−2
−1
0
1
2
3
4
2
0
−2
−4


D5000_linear — Train boundary (Perr=0.398)
Class 0
Class 1
x1
x2
−4
−2
0
2
4
4
2
0
−2
−4



Comment: The findings indicate that both the quantity of training samples and the form of the function have a substantial impact on classifier performance. The logistic-linear model, expressed as **$h(x, w) = 1 / (1 + e^{(-w^T z(x))})$** where **$z(x) = [1, x^T]^T$**, resulted in notably higher errors because its linear boundary was unable to effectively represent the curved class separation. In contrast, the logistic-quadratic model, utilizing **$z(x) = [1, x_1, x_2, x_1^2, x_1x_2, x_2^2]^T$**, obtained significantly reduced errors as it conformed to the non-linear class configuration. Expanding the training size (from 50 to 5000 samples) enhanced both models by decreasing estimation noise. The optimal quadratic model achieved P(error) = 0.278, almost the same as the Bayes-optimal error (0.28) from Part 1, validating that the quadratic logistic classifier can effectively approximate the theoretical minimum error with enough data present

Question – 2: Cubic Polynomial Regression with ML and MAP

This problem centers on estimating the weight vector w of a cubic polynomial that represents the connection between a two-dimensional input vector **$x = [x_1, x_2]^T$** and a scalar output **y**. The equation is expressed as **$y = c(x, w) + v$**, where **v** denotes additive Gaussian noise that has a mean of zero and a variance of **σ^2**. We aim to develop and apply Maximum Likelihood (ML) and Maximum a Posteriori (MAP) estimators for w and assess their effectiveness on a validation set, examining the influence of the MAP hyperparameter γ (gamma).

Theoretical Basis and Estimator Formulation:

To process the two-dimensional cubic polynomial, the input x is converted into an extended feature vector z(x) that encompasses all polynomial terms up to the third degree:

$$\mathbf{z(x) = [1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3]^T}$$

This transformation enables us to represent the intricate nonlinear relationship as a linear model in terms of parameters: **c(x, w) = w^T.z(x)**

Maximum Likelihood (ML) Estimation:
The ML estimator aims to find the parameter vector w that maximizes the likelihood of the given data. Assuming that the data is independent and identically distributed (i.i.d.) Gaussian noise is analogous to reducing the total of squared errors:

ŵ_ML = argmin_w Σ_{n=1}^N (y_n - w^T z(x_n))^2

The solution in closed form for this issue is provided by the normal equations:

ŵ_ML = (Z^T Z)^{-1} Z^T y

where Z represents the design matrix with the n-th row as **z(x_n)^T**, and **y** is the vector containing the target values. Although the ML estimator delivers an unbiased fit to the training data, it is susceptible to overfitting, especially in high-dimensional parameter spaces such as the one utilized here

Maximum a Posteriori (MAP) Estimation:
The MAP estimator includes a prior belief regarding the parameters, using a Bayesian method to minimize overfitting. We consider a Gaussian prior distribution for the weights: p(w) = N(w | 0, γ I), where I denotes the identity matrix. The MAP estimate is determined by optimizing the posterior distribution, which is the same as addressing the subsequent regularized minimization issue:

ŵ_MAP = argmin_w [Σ_{n=1}^N (y_n - w^T z(x_n))^2 + (σ^2/γ) ||w||^2]

This also results in a closed-form solution:

ŵ_MAP = (Z^T Z + (σ^2/γ) I)^{-1} Z^T y

The term **(σ^2/γ)** serves as a regularization factor, discouraging large weights and thus regulating model complexity. As **γ(gamma)** approaches infinity, the prior loses its informativeness and the MAP estimate aligns with the ML estimate. On the other hand, as **γ(gamma)** approaches 0, the prior pushes the weights towards zero

How is it implemented:
A synthetic dataset consisting of 200 samples was created, divided evenly into training and validation groups. The actual model was a cubic polynomial featuring fixed coefficients, and Gaussian noise was introduced to the outputs to mimic real-world measurement errors.

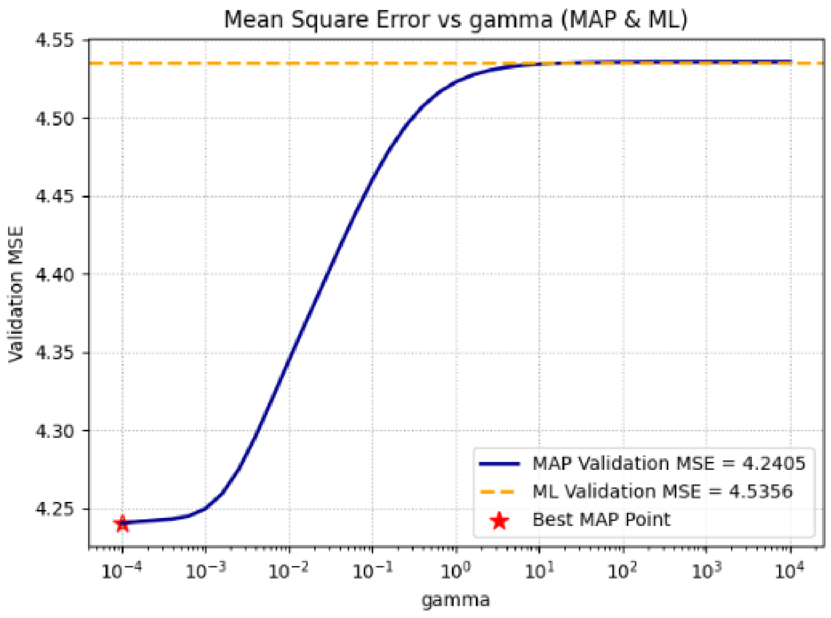
The execution continued in the following manner:

- **Feature Mapping**: The design matrix Z was created for both training and validation datasets by using the cubic feature transformation z(x) on every input sample.
- **ML Estimation**: The ML coefficients were calculated directly through the normal equations.
- **MAP Estimation**: The MAP weights were calculated for various γ values distributed logarithmically from 10^-4 to 10^4.
- **Assessment**: The effectiveness of every estimator was measured by computing the Mean Squared Error (MSE) on the validation dataset. The estimation error, characterized as the L2-norm of the disparity between the true and estimated weight vectors, was calculated as well

Results and Analysis:

```
RESULTS:
ML Estimator:
Mean Square Error = 4.535611
Estimation Error = 0.615125

MAP Estimator:
Best gamma = 0.000100
Mean Square Error = 4.240496
Estimation Error = 0.119639
```



The results from the experiments offer distinct understanding regarding how the estimators behave:

- **ML Estimator Effectiveness**: The ML estimator reached a MSE of 4.535611 with an estimation error of 0.615125. This elevated MSE suggests considerable overfitting, as the intricate model has captured the noise in the training data instead of the core function, resulting in weak generalization.
- **Performance of the MAP estimator and sensitivity to gamma**: The effectiveness of the MAP estimator was significantly influenced by the **γ (gamma)** value. The validation MSE in relation to **γ (gamma)** exhibited a typical U-shaped pattern.
    - With very low **γ (gamma)** values (intense regularization), the model performs poorly, leading to elevated MSE.
    - The best performance occurred at **γ (gamma)** = 0.0001, resulting in a notably lower validation MSE of 4.240496 and a considerably decreased estimation error of 0.119639.
    - With high **γ (gamma)** values (weak regularization), the MAP estimate aligns with the ML solution, and the MSE nears the ML baseline of 4.535611

The findings clearly show that the well-tuned MAP estimator outperforms the ML estimator in this situation. The ML estimator's elevated MSE (4.535611) is a typical indication of overfitting, which is anticipated due to the cubic model's high complexity.

The MAP estimator effectively reduced this overfitting by employing L2 regularization. The ideal **γ (gamma)** value of 0.0001 signifies a balance between the bias from the prior and the reduction in variance due to regularization. This is validated by the approximately 6.5% decrease in MSE (from 4.535611 to 4.240496) and the roughly 80% decline in parameter estimation error (from 0.615125 to 0.119639).

The significant decrease in estimation error demonstrates that the MAP estimator not only provided improved predictions but also retrieved a parameter vector significantly nearer to the actual underlying weights. This emphasizes an important advantage of regularization: it directs optimization towards a more reasonable and consistent solution. To sum up, for ill-posed issues such as this polynomial regression, the MAP estimator with suitable hyperparameter adjustment is crucial for obtaining models that are accurate and resilient

Question – 3: Vehicle Localization via MAP Estimation
The objective of this problem is to ascertain the most likely location of a vehicle, represented by the coordinates **x = [x, y]^T**, by merging imprecise distance readings from K known landmarks with our prior knowledge regarding the probable position of the vehicle. We tackle this by determining the Maximum A Posteriori (MAP) estimate, which represents the location that maximizes the posterior probability based on the data.

Based on Bayes' theorem, the posterior probability is related to the multiplication of the likelihood and the prior.
p(x | {r_i}) ∝ p({r_i} | x) * p(x)

The likelihood function, **p({r_i} | x)**, indicates the probability of achieving the measured ranges set **{r_1, ..., r_K}** given that the vehicle is located at position x. As the measurement noises are independent, the combined likelihood equals the product of the individual likelihoods for every landmark:

p({r_i} | x) = ∏_{i=1}^K p(r_i | x) = ∏_{i=1}^K N(r_i | d_i(x), σ_i^2)

where **d_i(x)** = ||**x - l_i**|| represents the actual distance from the candidate point x to the i-th landmark **l_i = [x_i, y_i]^T**, and N signifies the Gaussian distribution.

The prior probability, p(x), represents our initial assumption regarding the vehicle's position before obtaining any measurements. We consider a Gaussian prior with its center at the origin:

p(x) = N(x | 0, C) where the covariance matrix is **C = [[σ_x^2, 0], [0, σ_y^2]].**

Consequently, the MAP estimate is determined by resolving:
x_MAP = argmax_x [p(x) * ∏_{i=1}^K p(r_i | x)]

To simplify this issue for numerical optimization, we use the negative logarithm of the posterior. To minimize this negative log-posterior is to maximize the posterior. We can also eliminate any additive constants that are independent of x, since they do not influence the position of the minimum. Post-simplification, we derive the subsequent objective function, J(x), that we executed and minimized:

J(x) = (1/2)((x^2 / σ_x^2) + (y^2 / σ_y^2)) + ∑_{i=1}^K (1 / (2σ_i^2)) * (r_i - ||x - l_i||)^2

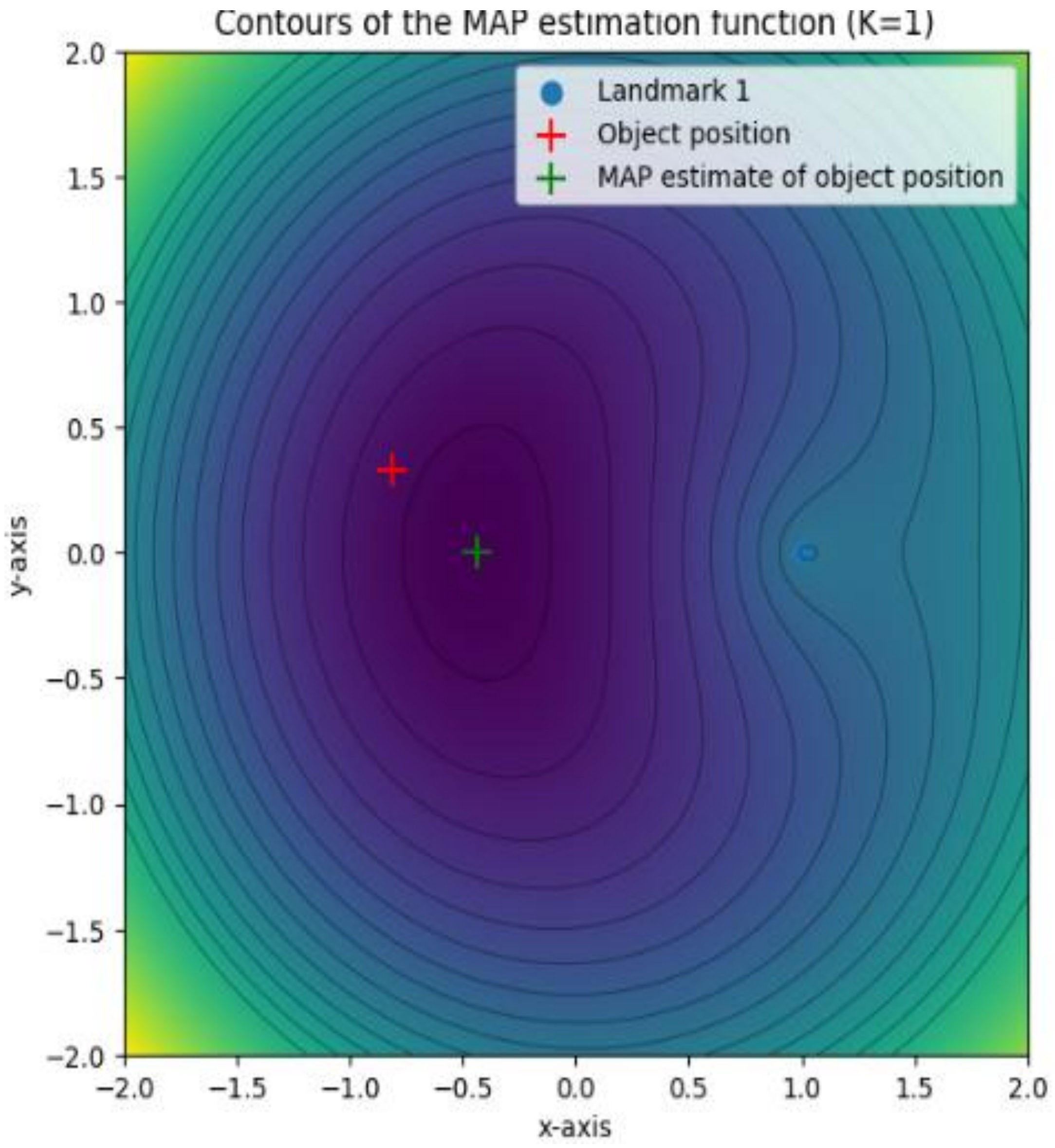
This function consists of two primary elements. The initial term, **(1/2)((x^2 / σ_x^2) + (y^2 / σ_y^2)),** discourages locations that are distant from the origin, representing our prior assumption. The second term**, ∑_{i=1}^K (1 / (2σ_i^2)) * (r_i - ||x - l_i||)^2**, represents a sum of squared errors that penalizes locations where the distances to the landmarks differ from the observed ranges. The MAP estimate is the x that reduces this overall cost function to its lowest point.

Implementation:
To test our MAP estimation framework, we set up a simulated scenario. The true position of the vehicle was fixed at [x_T, y_T]^T = [0.5, 0.3]^T, which lies inside the unit circle. For each experiment with a different number of landmarks K (specifically, K = 1, 2, 3, and 4), we placed the landmarks at evenly spaced intervals on the unit circle. For instance, for K=2, the landmarks were placed at [1, 0] and [-1, 0]; for K=4, they were placed at [1, 0], [0, 1], [-1, 0], and [0, -1].

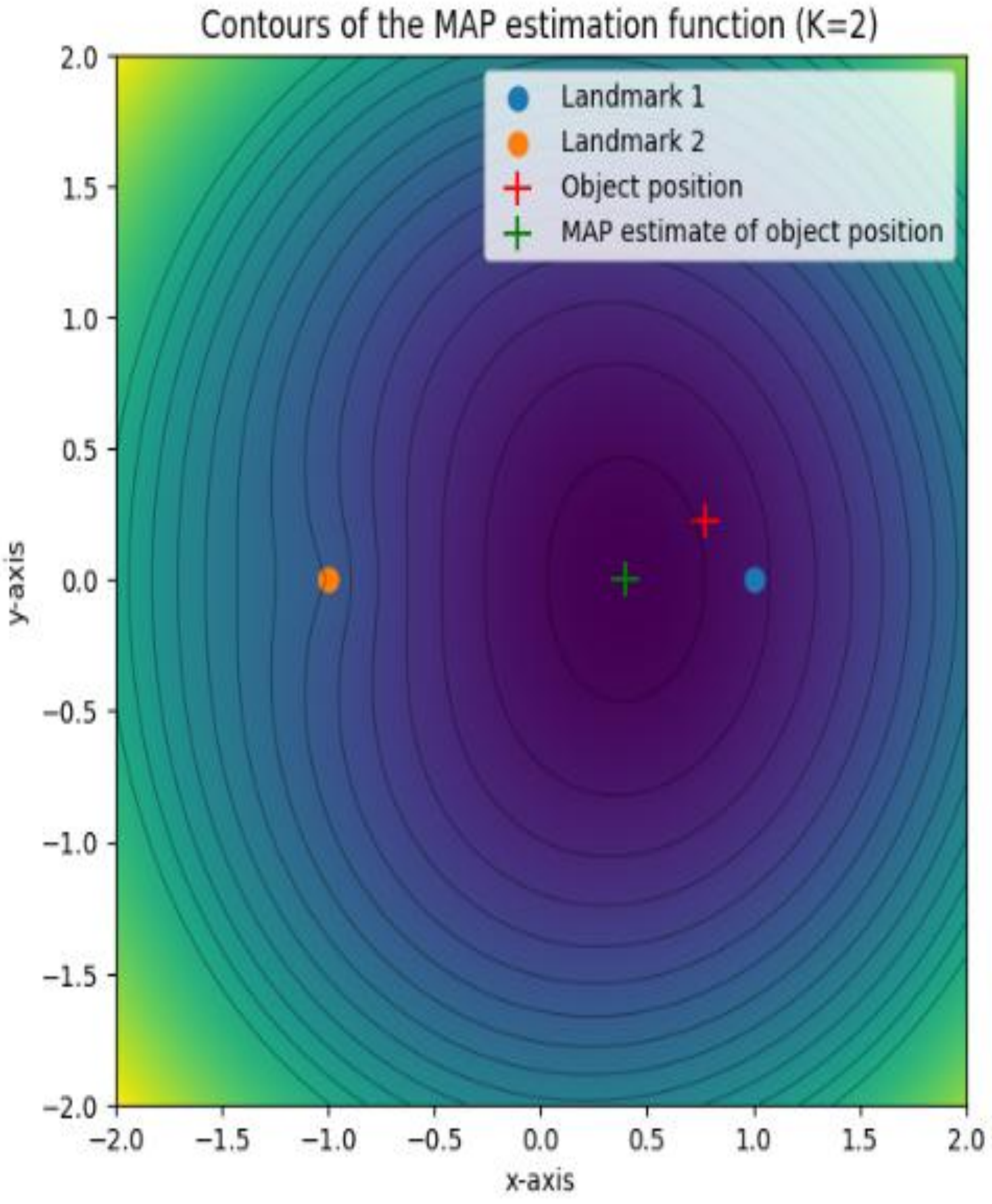
The range measurements were simulated by calculating the true distance from the vehicle to each landmark and then adding independent Gaussian noise with a standard deviation of σ_i = 0.3 for all i. If any simulated range measurement was negative, it was discarded and resampled to ensure physical realism. For the prior, we chose parameters σ_x = σ_y = 0.25, indicating a reasonably strong prior confidence that the vehicle is located near the origin.

The implementation in code followed a straightforward workflow for each value of K. First, the landmark positions were calculated. Then, the true distances were computed, and noisy range measurements were generated. The core of the implementation was defining the objective function J(x) as derived in the previous section. This function was then evaluated across a fine grid of x and y values, both ranging from -2.0 to 2.0. Finally, we plotted the equilevel contours of this function, which are lines connecting points of equal cost. The minimum point of this function, visible as the center of the innermost contour, represents the MAP estimate. For visualization, we superimposed the true vehicle position (marked with a '+') and the landmark positions (marked with an 'o') onto these contour plots.

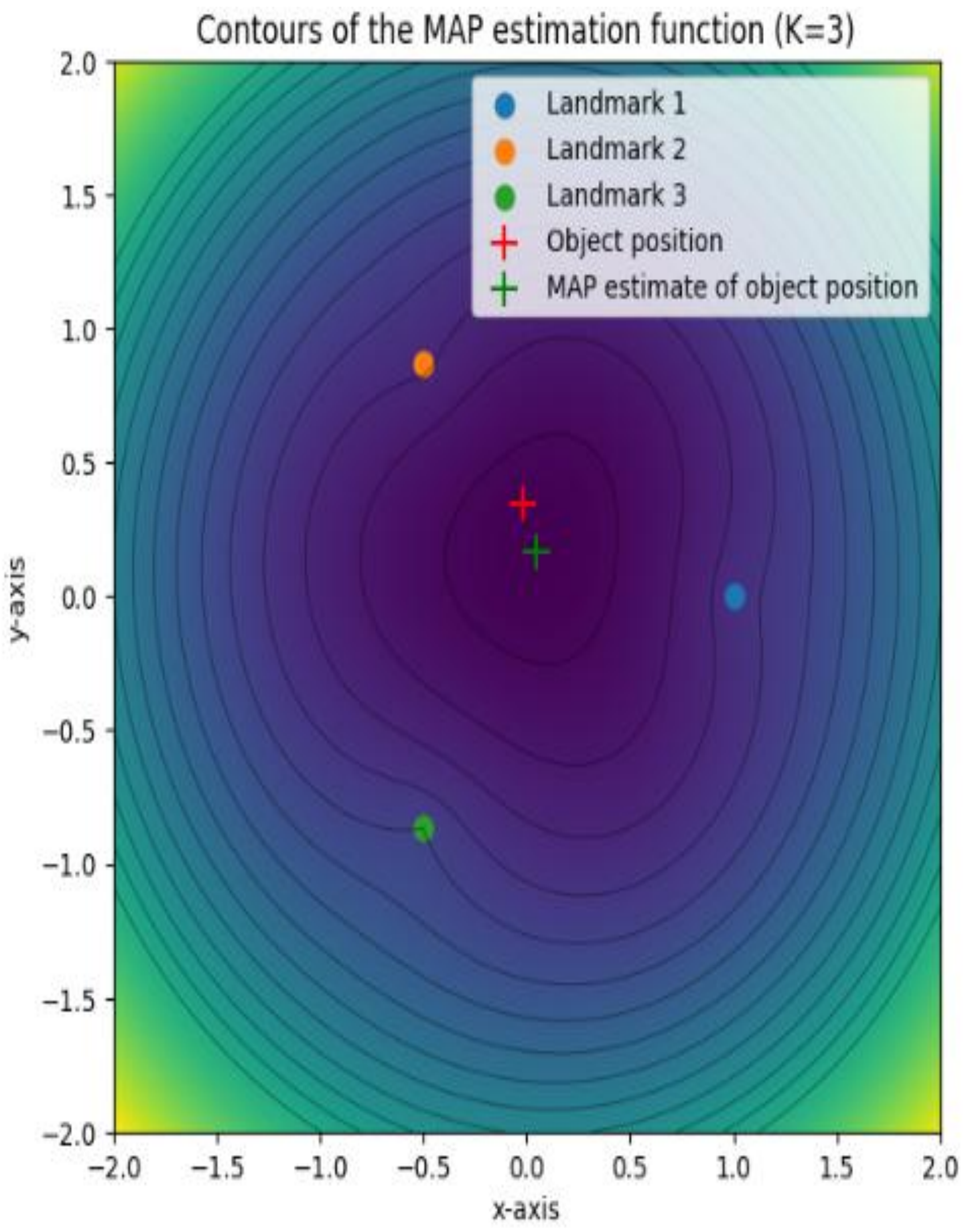
Results and Analysis:
The contour plots of the density functions, K = 1, 2, 3, 4, provide a clear visual narrative of how the estimate of the MAP evolves with more data.



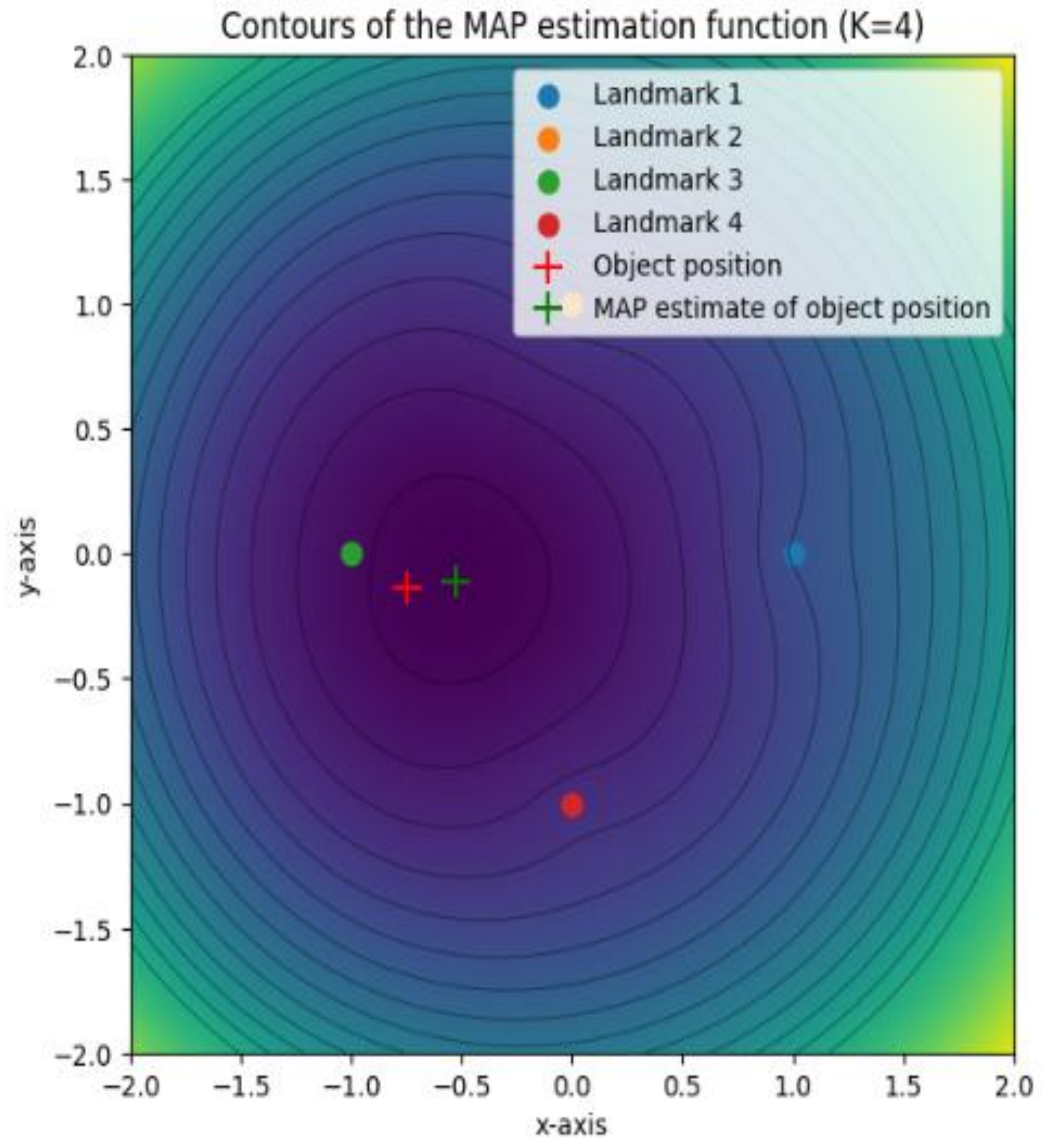
For K=1, the contour graph displays a broad, arched gorge. The vehicle's location is very uncertain in the direction tangent to the circle surrounding the lone landmark. The previous pulls the optimal estimate away from the actual position, leading to a MAP estimate that lacks accuracy.



With K=2, the two reference points create overlapping constraints. The boundaries create closed, stretched ellipses. The MAP estimate approaches the actual position more than in the K=1 scenario, yet the area of high uncertainty (the extensive zone within the outer contours) remains considerable, especially in the direction perpendicular to the line connecting the two landmarks.



Incorporating a third landmark (K=3) significantly enhances the situation. The shapes turn increasingly circular and are closely clustered around a particular point. The MAP estimate is now nearly identical to the actual vehicle location, and the region of uncertainty has been significantly decreased. This indicates that the three measurements offer strong limits in all directions.



Ultimately, with K=4, the contours are quite close, nearly perfect concentric circles centered around the actual position. The MAP estimate is almost identical to the actual location, signifying that the vehicle's position has been identified with great precision and accuracy

Comment: The contour plots support two conclusions strongly. The first is that as K increases, the MAP estimate seems to converge towards the true location. The estimation is able to do this, since we have only one or two reference points, and thus is overloaded by the prior and constrained by the amount of data. As we start incorporating the measurements of three and eventually four landmarks, the estimate converges towards the true location. This is because as we gather information from multiple sensors or other sources, we are progressively able to beat the initial prior belief from scratch.

The second conclusion is that as K increases, we become much more certain of our estimate. Also quite obvious from the contour plots themselves. Moving from K=1 and to K=4, the contours are packed closer together, as well as the innermost contour decreasing massively in size. A narrower contour plot means the posterior probability distribution over the possible location is tighter, so the estimate has less variance and we are more confident. In practice, each additional landmark gives us another independent constraint about the possibilities of where the vehicle is located, therefore reducing uncertainty. The whole thing reminds one of the process of sensor fusion, that proper addition of multiple sources of information involving imperfect elements will result in something that is both more precise and more certain than that which can be extracted from any single source.

Question – 4:

Question 4:

It is a classification problem ~~where~~ with

c classes: $\omega_1, \omega_2, \ldots \omega_c$

c+1 actions: classify as class 1, 2 ...., c or reject

Loss function:

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

Step 1: For any action α_i, the conditional risk is:

$$R(\alpha_i \mid x) = \sum_{j=1}^{c} \lambda(\alpha_i \mid \omega_j) . P(\omega_j \mid x)$$

Step 2: Consider the risk for classification as class i where ~~class~~ $i \in \{1, 2 \ldots \ldots c\}$

$$R(a_i \mid x) = \lambda(\alpha_i \mid \omega_i) . P(\omega_i \mid x) + \sum_{j \neq i} \lambda(\alpha_i \mid \omega_j) . P(\omega_j \mid x)$$

Substitute the loss values:

→ when $j = 1$; loss $= 0$ (correct classification)

→ when $j \neq i$; loss $= \lambda_s$ (misclassification)

$$R(\alpha_i \mid x) = 0 \times P(\omega_i \mid x) + \lambda_s \cdot \sum_{j \neq i} P(\omega_j \mid x)$$

~~Since $\sum_j$~~ (P.

Since $\sum_j P(\omega_j \mid x) = 1$, we have

$$\sum_{j \neq i} P(\omega_j \mid x) = 1 - P(\omega_i \mid x)$$

$$R(\alpha_i \mid x) = \lambda_s \left(1 - P(\omega_i \mid x)\right)$$

Step-3: Now we have to calculate risk of rejection

$$R(\alpha_{c+1} \mid x) = \sum_{j=1}^{c} \lambda(\alpha_{c+1} \mid \omega_j) \cdot P(\omega_j \mid x)$$

Since $\lambda(\alpha_{c+1} \mid \omega_j) = \lambda_r$ for all j

$$R(\alpha_{c+1} \mid x) = \lambda_r \sum_{j=1}^{c} P(\omega_j \mid x)$$

$$= \lambda_r \cdot 1 = \lambda_r$$

Step 4: This step is to determine when to classify or reject.

→ when $R(\alpha i \mid x) < R(\alpha c+1 / x)$

Substituting the risk expressions:

$$\lambda_s(1 - P(w_i \mid x)) < \lambda_r$$

$$\lambda_s - \lambda_s P(w_i \mid x) < \lambda_r$$

$$\lambda_s - \lambda_r < \lambda_s . P(w_i / x)$$

When we divide both sides by λ_s:

$$1 - \frac{\lambda_r}{\lambda_s} < P(w_i \mid x)$$

ep 5: This step is to determine which class to choose

Among all classes, we should choose one with highest posterior probabilites

$i = \text{argmax } P(w_i | x)$

step 6: This step is about Optimal Decision Rule

Decide w_i if:

→ $P(w_i | x) \geq P(w_j | x)$ for all $j \in \{1, 2, \dots c\}$

→ $P(w_i | x) \geq 1 - \frac{\lambda_r}{\lambda_s}$

Special cases:

(1) Case - 1: when $\lambda_r = 0$

Threshold becomes:

$$P(w_i | x) \geq 1 - 0/\lambda_s = 1$$

Which means we must be 100% certain to classify, if otherwise then reject.

(2) Case 2 : When $\lambda_r > \lambda_s$
Threshold becomes :

$$P(w_i|x) \geq 1 - \lambda_r/\lambda_s < 0$$

which means any probability $P(w_i|x)$ greater than zero satisfies this. It would be better to pick the most likely class. Making an error is better than rejecting.

Question – 5:

Question-5:

Given

① Random Variable: $Z \sim Cat(\theta)$

② $Z = [z_1, z_2, z_3 \ldots z_K]^T$

$Z_K = 1$ if it is in state k

$Z_K = 0$ otherwise

③ $\theta = [\theta_1, \theta_2 \ldots \theta_K]^T$ where

$P(Z_K = 1) = \theta_K$

④ Constraint: $\sum_{K=1}^{K} \theta_K = 1$

~~⑤ Data: D =~~

Part-1: Maximum Likelihood (ML) Estimator

Step-1: Probability of one sample

For a single sample Z_n:

$$P(Z_n | \theta) = \prod_{k=1}^{K} \theta_k^{(Z_{nk})}$$

step 2: Likelihood for all N samples

Since samples are identical and

independent distributed

$$P(D|\theta) = \prod_{n=1}^{N} P(z_n|\theta)$$

$$= \prod_{n=1}^{N} \prod_{k=1}^{K} \theta_k^{z_{nk}}$$

$$\Rightarrow P(D|\theta) = \prod_{k=1}^{K} \theta_k^{\sum_{n=1}^{N} z_{nk}}$$

If $N_k = \sum_{n=1}^{N} z_{nk}$

$$P(D|\theta) = \prod_{k=1}^{K} \theta_k^{N_k}$$

Step 3: Take the log likelihood

$$\log P(D|\theta) = \log \left[\prod_{k=1}^{K} \theta_k^{N_k}\right]$$

$$\log P(D|\theta) = \sum_{k=1}^{K} N_k \log \theta_k$$

Step 4: Constraint optimization setup

We have to maximize $\log P(D|\theta)$
subject to $\sum_k \theta_k = 1$

Use Lagrange multiplier λ:

$$L(\theta, \lambda) = \sum_{k=1}^{K} N_k \log \theta_k + \lambda\left(1 - \sum_{k=1}^{k} \theta_k\right)$$

step5: Take derivative w.r.t θ_k

$$\frac{\partial L}{\partial \theta_k} = \frac{N_k}{\theta_k} - \lambda = 0$$

Solve for θ_k:

$$\theta_k = \frac{N_k}{\lambda}$$

Step6: Constraint is used to find λ

Substitute into constraint

$$\sum_k \theta_k = 1:$$

$$\sum_{k=1}^{K} \left(\frac{N_k}{\lambda}\right) = 1 \Rightarrow \left(\frac{1}{\lambda}\right) \sum_{k=1}^{K} N_k = 1$$

Since $\sum_k N_k = N$ (total Samples):

$$\frac{N}{\lambda} = 1 \Rightarrow \lambda = N$$

Step 7: Result of ML estimator

Substitute $\lambda = N$ in θ_k:

$$\theta_k(ML) = \frac{N_k}{N}$$

Part-2: MAP Estimator with Dirichlet Prior

Step 1: Prior Distribution equation

$$P(\theta|\alpha) = (1/B(\alpha)) \cdot \prod_{k=1}^{k} \theta_k^{\alpha_k - 1}$$

where $\alpha = [\alpha_1, \alpha_2 --- \alpha_k]^T$ are hyperparameters & $B(\alpha)$ is a normalization constant.

Step 2: Write the Posterior using Bayes Theorem

$$P(\theta|D, \alpha) \propto P(D|\theta) \cdot P(\theta|\alpha)$$

Substituting the above equations we get

$$P(\theta|D, \alpha) \propto \left[\prod_{k=1}^{k} \theta_k^{N_k}\right] \cdot \left[\prod_{k=1}^{k} \theta_k^{\alpha_k - 1}\right]$$

Combining the above exponents we get:

$$P(\theta|D, \alpha) \propto \prod_{K=1}^{K} \theta_K^{(N_K+\alpha_K-1)}$$

Step 3: Take the log posterior

$$\log P(\theta|D, \alpha) = \sum_{K=1}^{K} (N_K+\alpha_K-1) \log \theta_K + \text{constant}$$

Step 4: Constraint optimization setup

Log posterior is maximized to $\sum_K \theta_K = 1$

$$\mathcal{L}(\theta, \lambda) = \sum_{K=1}^{K} (N_K+\alpha_K-1) \log \theta_K + \lambda\left(1 - \sum_{K=1}^{K} \theta_K\right)$$

Step 5: Derivative w.r.t θ_K

$$\frac{\partial \mathcal{L}}{\partial \theta_K} = \frac{(N_K+\alpha_K-1)}{\theta_K} - \lambda = 0$$

Solve for θ_k:

$$\theta_k = \frac{N_k + \alpha_k - 1}{\lambda}$$

step 6: Use the constraint to find λ

$$\sum_{k=1}^{K} \theta_k = 1 \Rightarrow \sum_{k=1}^{k} \frac{(N_k + \alpha_k - 1)}{\lambda} = 1$$

$$\lambda = \sum_{k=1}^{k} (N_k + \alpha_k - 1)$$

$$\lambda = \sum_k N_k + \sum_k \alpha_k - K$$

$$= \cancel{N_k}\ N + \sum_k \alpha_k - K$$

step 7: Result of MAP Estimator

$$\theta_K^{MAP} = \frac{(N_K + \alpha_K - 1)}{(N + \sum_{j=1}^{K} \alpha_j - K)}$$

The link to the online repository of the codes:
https://github.com/PRAYAG2000n/Intro_to_ML_EECE5644/tree/main/Assignment%20-%202

Citations:

1. A Python implementation of linear & logistic regression:
   Linear-and-Logistic-Regression-Algorithm-Using-Python by rposhal:
   https://github.com/rposhala/Linear-and-Logistic-regression-Algorithm-using-Python

2. A demonstration of ridge regression with Python (akin to MAP with ℓ_2 prior):
   Ridge-Regression-in-Python by margaritageleta:
   https://github.com/margaritageleta/ridge-regression-python

3. A toy 2D SLAM (localization + landmarks) implementation: slam_2d by mxagar:
   https://github.com/mxagar/slam_2d

4. A project for landmark localization: Landmark-Detection-Localization by abedinsherifi.
   https://github.com/abedinsherifi/Landmark-Detection-Localization

5. Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern Classification* (2nd ed.). John Wiley & Sons.

6. Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.