EE 583 Pattern Recognition HW2

Eda Özkaynar 2375582

QUESTION 1 MAXIMUM LIKELIHOOD ESTIMATION FROM SAMPLE DATA

Maximum-Likelihood

Let $X = \{x_1, x_2, \ldots, x_n\}$ be our dataset. We define the class conditional density function as $p(X|\theta)$. Thus:

$$\hat{\theta}_{ML} = \arg\max_{\theta} p(X|\theta)$$

Taking the logarithm, we obtain:

$$\hat{\theta}_{ML} = \arg\max_{\theta} \log p(X|\theta)$$

Assuming the distribution is i.i.d.

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

$$\hat{\theta} = \arg\max_{\theta} \log p(X|\theta) = \arg\max_{\theta} \log \prod_{k} p(x_k|\theta) = \arg\max_{\theta} \sum_{k} \log p(x_k|\theta)$$

$$p(x_i|\Theta) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x_i} - \mu)^{\mathbf{T}}\mathbf{\Sigma^{-1}}(\mathbf{x_i} - \mu)\right) \quad \text{where } d = 2$$

For a Gaussian distribution, the log-likelihood function becomes:

$$-\log p(x_i|\Theta) = -\frac{1}{2}\log 2\pi - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(\mathbf{x_i} - \mu)^{\mathbf{T}}\mathbf{\Sigma^{-1}}(\mathbf{x_i} - \mu)$$

$$\text{where } \Theta = \begin{bmatrix} \mu \\ \Sigma \end{bmatrix}$$

Taking gradient,

$$\nabla_{\Sigma}\left(\log p(x_i|\Theta)\right) = \begin{bmatrix} \Sigma^{-1}(\mathbf{x_i} - \mu) \\ -\frac{1}{2}\Sigma^{-1} - \frac{1}{2}\Sigma^{-1}(\mathbf{x_i} - \mu)(\mathbf{x_i} - \mu)^{\mathbf{T}}(*) \end{bmatrix}$$

Mean estimator

$$\sum_{i=1}^{N} \Sigma^{-1}(\mathbf{x_i} - \mu) = \mathbf{0}$$

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

Variance estimator,

$$\sum_{i=1}^{N}\left(-\frac{1}{2}\Sigma^{-1} - \frac{1}{2}\Sigma^{-1}(\mathbf{x_i} - \mu)(\mathbf{x_i} - \mu)^{\mathbf{T}}\right) = 0$$

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

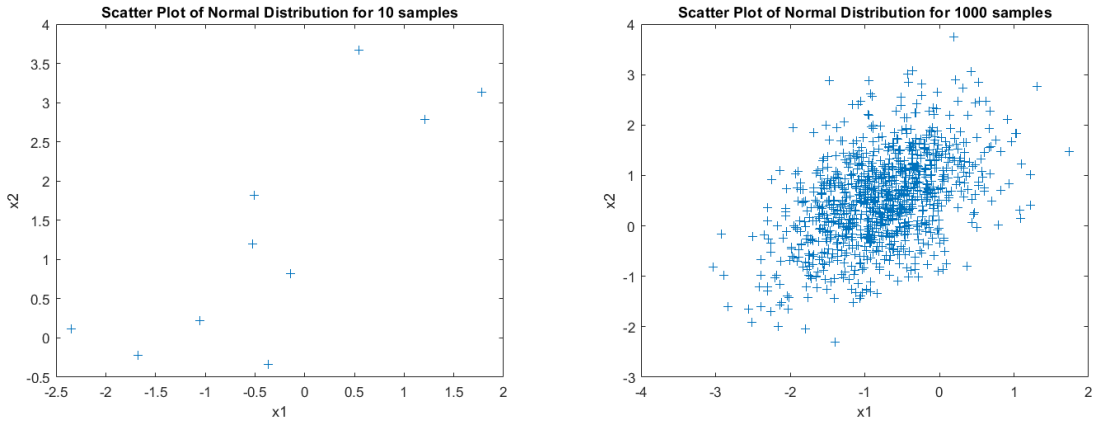
$$\hat{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x_i} - \mu)(\mathbf{x_i} - \mu)^{\mathbf{T}}$$

Proof of (*)

$$\alpha = \mathbf{x^T \Sigma^{-1} x} \quad \alpha \text{ is scalar}$$

$$\alpha = \sum_{i=1}^{2}\sum_{j=1}^{2} x_i a_{ij} x_j$$

$$\nabla_\Sigma \alpha = \begin{bmatrix} \frac{\partial \alpha}{\partial a_{11}} & \frac{\partial \alpha}{\partial a_{12}} \\ \frac{\partial \alpha}{\partial a_{21}} & \frac{\partial \alpha}{\partial a_{22}} \end{bmatrix} = \begin{bmatrix} x_1^2 & x_1 x_2 \\ x_2 x_1 & x_2^2 \end{bmatrix} = \mathbf{x x^T}$$



(a) Plot of 10 samples

(b) Plot of 1000 samples

Fig. 1: Plot of samples from distribution

MATLAB outputs are,

$$\hat{\mu}_{10} = \begin{bmatrix} -0.3086 \\ 1.3199 \end{bmatrix} \quad \hat{\mu}_{1000} = \begin{bmatrix} -0.7825 \\ 0.5143 \end{bmatrix}$$

$$\hat{\Sigma}_{10} = \begin{bmatrix} 1.4096 & 1.3343 \\ 1.3343 & 1.9229 \end{bmatrix} \quad \hat{\Sigma}_{1000} = \begin{bmatrix} 0.4811 & 0.2839 \\ 0.2839 & 0.7896 \end{bmatrix}$$

In Figure 1(a), the 10-sample plot appears sparse making it difficult to discern the overall shape of the distribution. In Figure 1(b), the 1000-sample scatter plot better illustrates the structure of a Gaussian distribution, showing a dense, elliptical pattern that indicates the covariance structure of the data. Since the mean μ is calculated as the average of the data points and the covariance matrix Σ is derived based on the deviations of the data points from the mean, as the sample size increases the estimators converge to the true parameters and provide more reliable estimates. As we can see from the outputs, the number of samples increases the estimator values, which are getting closer to the real values.

QUESTION 2 BAYESIAN PARAMETER ESTIMATION

A. *i*

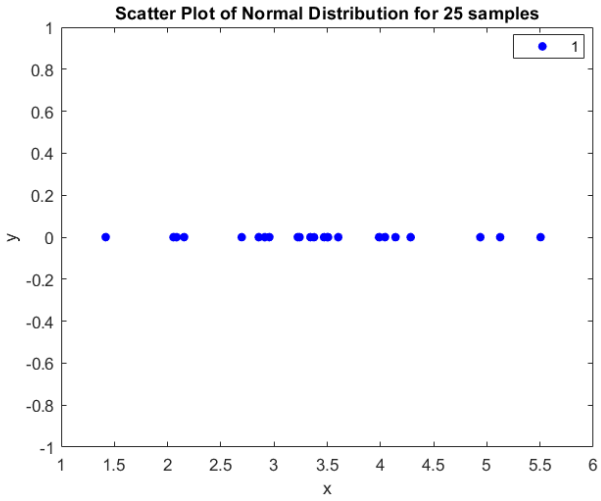


Fig. 2: Plot of 25 samples

$$\hat{\mu}_{ML} = 3.4366$$

B. *ii.*

For maximum a posterior probability estimation,

$$\hat{\mu}_{MAP} = \frac{n\sigma_\mu^2}{n\sigma_\mu^2 + \sigma^2} \cdot \frac{1}{n}\sum_{i=1}^{n} x_i + \frac{\sigma^2}{n\sigma_\mu^2 + \sigma^2}\,\mu_0 \tag{1}$$

Equation 1 is a weighted sum of information coming from data and information coming from prior. Suppose data has fewer samples, estimation is based on prior information. As the number of samples increases, estimation relies on the information coming from data most. The MATLAB result is,

$$\hat{\mu}_{ML} = 3.4177$$

The MAP result is slightly better than the ML result. This is because the estimate is also based on prior knowledge. Since the sample size of 25 is relatively small, both the data and prior knowledge are utilized.

C. *iii.*

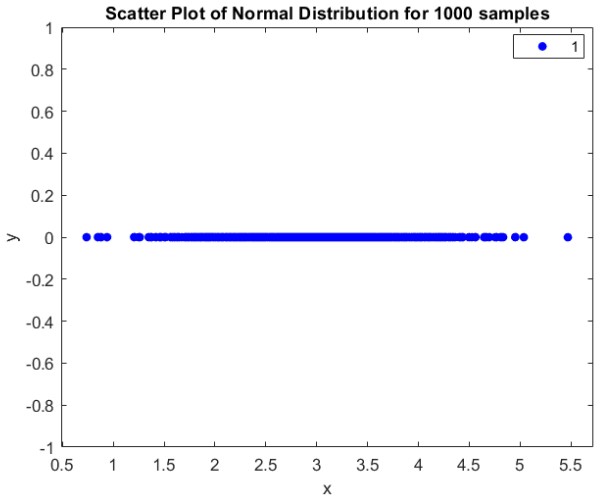
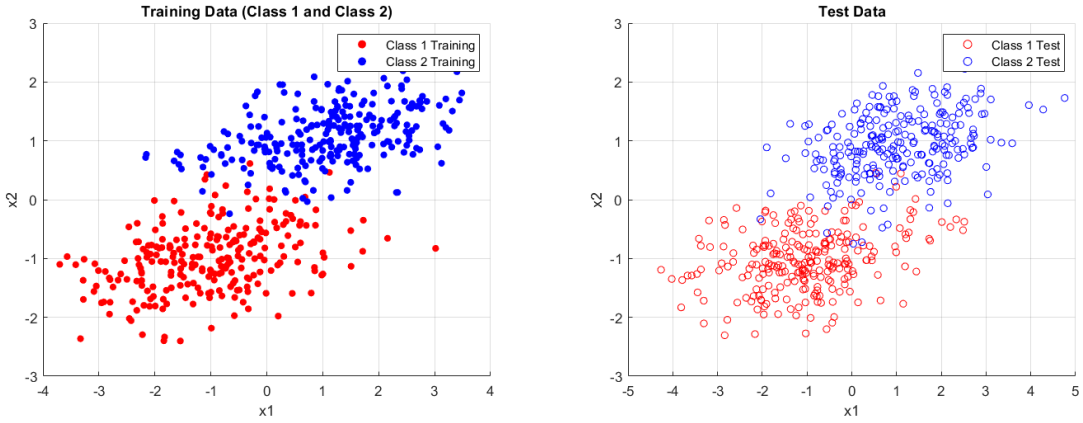


Fig. 3: Plot of 1000 samples

$$\hat{\mu}_{ML} = 2.9650 \quad \hat{\mu}_{MAP} = 2.9649$$

As the number of samples increases, the results of the estimator are closer to the true mean value. However, the result of MAP estimator is slightly less than the result of ML, this is because it uses prior knowledge whose mu value is 2.8, so the prior mu value affects the result of MAP estimator.

QUESTION 3 MINIMUM ERROR-RATE CLASSIFIER



(a) (b)

Fig. 4: Plots of train and test sets

D. i.

Minimum Risk Classifier

$$R\left(\alpha_i|\mathbf{x}\right) = \sum_j \lambda\left(\alpha_i|\omega_j\right) P\left(\omega_j|\mathbf{x}\right)$$

Minimum Error-Rate Classifier

$$\lambda\left(\alpha_i|\omega_j\right) = \begin{cases} 1 & \text{if } i \neq j \\ 0 & \text{else} \end{cases}$$

Therefore,

$$R\left(\alpha_i|\mathbf{x}\right) = \sum_{i\neq j} P\left(\omega_j|\mathbf{x}\right) \quad \Rightarrow \quad \text{To minimize risk, one should maximize the posterior probability.}$$

Discriminant Function

$$g_i(\mathbf{x}) = P(\omega_i|\mathbf{x}) = P(\mathbf{x}|\omega_i)P(\omega_i)$$

$$= \ln P(\mathbf{x}|\omega_i) + \ln P(\omega_i) \quad \text{(since prior probabilities are equal)}$$

$$P(\mathbf{x}|\omega_i) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_\mathbf{i})^\mathbf{T}\mathbf{\Sigma^{-1}}(\mathbf{x}-\mu_\mathbf{i})\right), \quad \text{where } d = 2$$

$$\ln P(\mathbf{x}|\omega_i) = -\frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma| - \frac{1}{2}(\mathbf{x}-\mu_\mathbf{i})^\mathbf{T}\mathbf{\Sigma^{-1}}(\mathbf{x}-\mu_\mathbf{i})$$

$$\Sigma_i = \Sigma_j = \Sigma$$

Omitting constants,

$$g_i(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mathbf{x} + \mu_\mathbf{i}^\mathbf{T}\mathbf{\Sigma^{-1}}\mathbf{x} - \frac{\mathbf{1}}{\mathbf{2}}\mu_\mathbf{i}^\mathbf{T}\mathbf{\Sigma^{-1}}\mu_\mathbf{i}$$

since $\mathbf{x}^T\Sigma^{-1}\mathbf{x}$ is common for both classes,

$$g_i(\mathbf{x}) = \mu_\mathbf{i}^\mathbf{T}\mathbf{\Sigma^{-1}}\mathbf{x} - \frac{\mu_\mathbf{i}^\mathbf{T}\mathbf{\Sigma^{-1}}\mu_\mathbf{i}}{\mathbf{2}}$$

$$g_j(\mathbf{x}) = \mu_\mathbf{j}^\mathbf{T}\mathbf{\Sigma^{-1}}\mathbf{x} - \frac{\mu_\mathbf{j}^\mathbf{T}\mathbf{\Sigma^{-1}}\mu_\mathbf{j}}{\mathbf{2}}$$

Decision Boundary

$$g_i(\mathbf{x}) = g_j(\mathbf{x})$$

$$\mu_\mathbf{i}^\mathbf{T}\mathbf{\Sigma^{-1}}\mathbf{x} - \frac{\mathbf{1}}{\mathbf{2}}\mu_\mathbf{i}^\mathbf{T}\mathbf{\Sigma^{-1}}\mu_\mathbf{i} = \mu_\mathbf{j}^\mathbf{T}\mathbf{\Sigma^{-1}}\mathbf{x} - \frac{\mathbf{1}}{\mathbf{2}}\mu_\mathbf{j}^\mathbf{T}\mathbf{\Sigma^{-1}}\mu_\mathbf{j}$$

$$\mu_\mathbf{i}^\mathbf{T}\mathbf{\Sigma^{-1}}\mathbf{x} - \mu_\mathbf{j}^\mathbf{T}\mathbf{\Sigma^{-1}}\mathbf{x} = -\frac{\mathbf{1}}{\mathbf{2}}\mu_\mathbf{i}^\mathbf{T}\mathbf{\Sigma^{-1}}\mu_\mathbf{i} + \frac{\mathbf{1}}{\mathbf{2}}\mu_\mathbf{j}^\mathbf{T}\mathbf{\Sigma^{-1}}\mu_\mathbf{j}$$

$$(\mu_\mathbf{i} - \mu_\mathbf{j})^\mathbf{T}\mathbf{\Sigma^{-1}}\mathbf{x} = \frac{\mathbf{1}}{\mathbf{2}}\mu_\mathbf{i}^\mathbf{T}\mathbf{\Sigma^{-1}}\mu_\mathbf{i} - \frac{\mathbf{1}}{\mathbf{2}}\mu_\mathbf{j}^\mathbf{T}\mathbf{\Sigma^{-1}}\mu_\mathbf{j}$$

$$(\mu_\mathbf{i} - \mu_\mathbf{j})^\mathbf{T}\mathbf{\Sigma^{-1}}\left(\mathbf{x} - \frac{\mu_\mathbf{i} + \mu_\mathbf{j}}{\mathbf{2}}\right) = \mathbf{0}$$

This represents a hyperplane through $\frac{\mu_\mathbf{i}+\mu_\mathbf{j}}{2}$.

(a) (b)

Fig. 5: Plot of Test and Train data with decision boundary using known parameters.

(a) (b)

Fig. 6: Plot of Test and Train data with decision boundary using ML estimator results.

The decision boundaries are linear due to equal covariances and the error rates show that both known and ML-estimated parameters perform comparably well. This suggests that ML estimation is a robust method for parameter estimation when the true parameters are unknown. However, ML parameter estimation is based on the training set, so the error rate on the test set can be large.

QUESTION 4 NON-PARAMETRIC DENSITY ESTIMATION

$$k_{\text{obs}} = \mathbb{E}[k] = n\,p \quad \Rightarrow \quad p = \frac{k_{\text{obs}}}{n}$$

$$p = \int_R p(x)\,dx^* \approx p(x)\,V \quad \Rightarrow \quad p(x) \approx \frac{p}{V} = \frac{k_{\text{obs}}}{Vn}$$

Parzen: Initial V_0 volume shrinking

$$V_n = \frac{V_0}{\sqrt{n}}$$

$$\text{Window function} \quad \Phi(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}$$

$$p_n(\vec{x}) = \frac{1}{n}\sum_{k=1}^{n} \frac{1}{V_n} \Phi\left(\frac{\vec{x} - \vec{x}_k}{h_n}\right)$$

Assume there exists a cube,

$$V_0 = h_1^3$$

$$V_n = \frac{V_0}{\sqrt{n}}$$

$$h_n = (V_n)^{1/3}$$

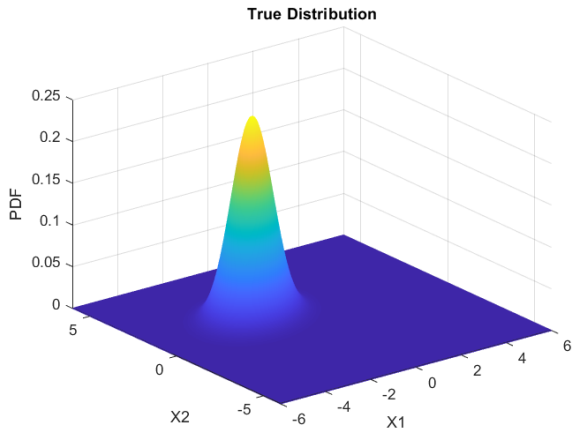
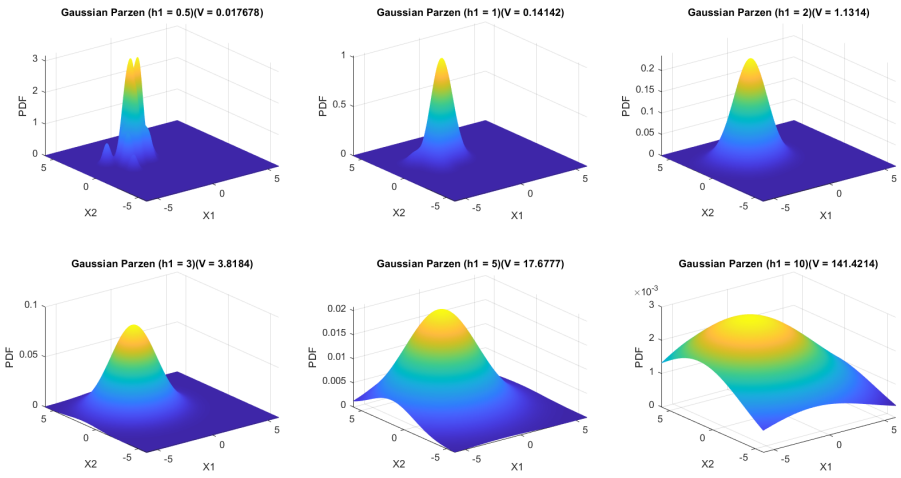
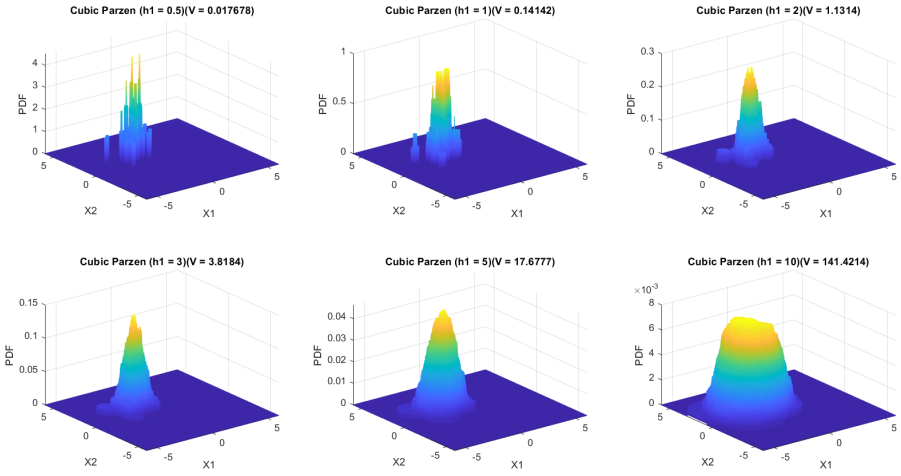


Fig. 7: Plot of true distribution



(a) Gaussian Parzen window



(b) Cubic Parzen window

Fig. 8: Parzen window estimations for different initial volumes

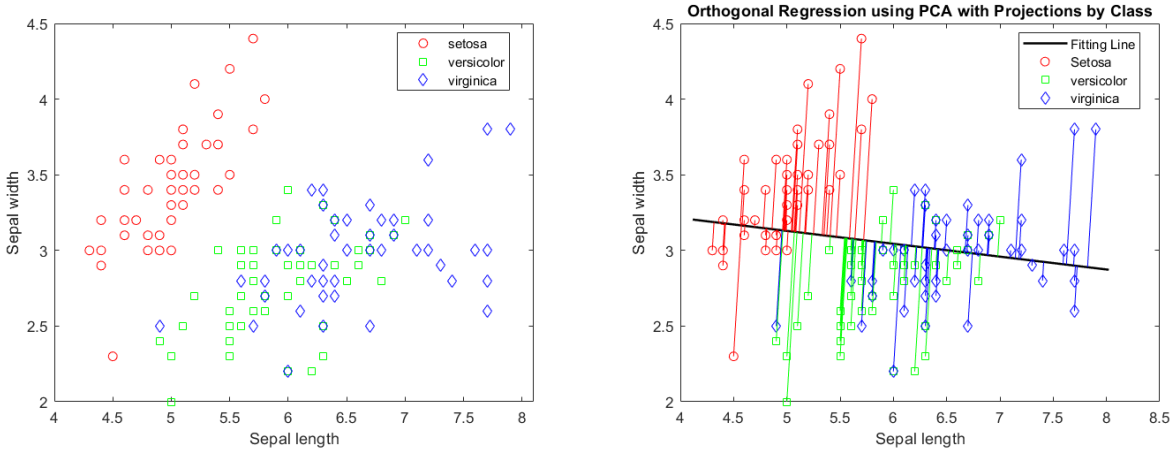
As can be seen in Figure 8, when h is small (V is small), the density estimates are highly localized, resulting in sharp, narrow peaks. This reflects a particular view of the data. Moreover, the model captures fine details and may lead to overfitting because it responds closely to individual points rather than the overall data distribution.

As h increases, the density estimate begins to smooth out, capturing broader trends rather than focusing on specific points.

For large h values, the density estimate becomes overly smooth. The model generalizes heavily, and individual data points have minimal influence on the shape of the density. This can lead to underfitting.

The Gaussian window provides smoother and more continuous estimates, while the cubic window maintains a segmented, less smooth appearance. This is because the Gaussian window adds the effect of all samples.

QUESTION 5 FITTING AN ORTHOGONAL REGRESSION USING PCA



(a) Scatter plot of 3 classes

(b) Fitting a Line to 2-D Data

Fig. 9

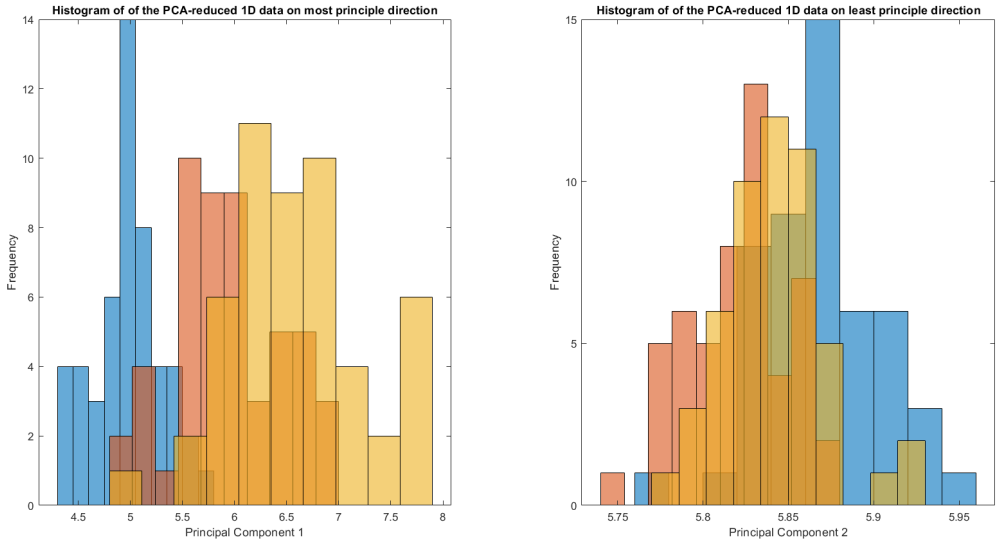


Fig. 10: Histograms of 1D data

In Figure 9a, this scatter plot visualizes the distribution of three iris classes (setosa, versicolor, virginica) in a 2D space based on Sepal Length (x-axis) and Sepal Width (y-axis). The plot shows that Setosa is relatively distinct from the other two classes, with minimal overlap. Versicolor and Virginica exhibit some overlap, indicating that these two classes are not easily separable based on just these two features. In the plot 9b, Principal Component Analysis (PCA) is used to fit an orthogonal regression line to the data, capturing the direction of maximum variance (the first principal component). The black line represents the principal component axis, serving as the best-fit line for the data. Each data point has a vertical projection line connecting it to the principal component line, illustrating the orthogonal distances.

As can be seen in Figure **??**, the first plot is represented by the first principal component (PC1), which captures the maximum variance in the data. It provides the direction along which the data points are most spread out. Score(:,1) corresponds to the projections along this direction. The most principal direction is indeed the direction of the largest eigenvalue, as this eigenvalue indicates the principal component that captures the most information (variance) about the data distribution. The second principal component (PC2) captures the second-highest variance, orthogonal to the first. It typically contains less variation than PC1, but it can still provide useful information about the structure of the data. score(:,2) gives projections along this direction Setosa shows a fairly distinct distribution with minimal overlap with the other two classes. Versicolor and Virginica exhibit significant overlap in their distributions along PC1, PCA may not fully separate these two classes.

```
clear; clc; close all;

%% Question 1 %% Maximum Likelihood

rng('default')  % For reproducibility

mu1     = [-.75 .5];
Sigma1  = [.5 .3 ; .3 .8];


X_10 = mvnrnd(mu1,Sigma1,10);

% % Visualize the data
figure;
plot(X_10(:,1),X_10(:,2),'+')
title('Scatter Plot of Normal Distribution for 10 samples');
xlabel('x1')
ylabel('x2')

% Maximum likelihood estimation for 10 samples

N_10               = size(X_10,1);      % Number of samples
mean_estimator_10  = sum(X_10,1)/N_10;  % Summing along the first dimension (rows)

% Initialize outer product
outer_product1     = zeros(size(2, 2));  % Initialize to the correct size

for i=1:N_10
    outer_product1 = outer_product1 + (transpose(X_10(i,:)) - transpose↙
(mean_estimator_10))*transpose(transpose(X_10(i,:)) - transpose↙
(mean_estimator_10));
end

varience_estimator_10 = outer_product1/N_10;

% Maximum likelihood estimation for 1000 samples
X_1000 = mvnrnd(mu1,Sigma1,1000);

% Visualize the data
figure;
plot(X_1000(:,1),X_1000(:,2),'+')
title('Scatter Plot of Normal Distribution for 1000 samples');
xlabel('x1')
ylabel('x2')

N_1000               = size(X_1000,1);    % Number of samples
mean_estimator_1000  = sum(X_1000,1)/N_1000;   % Summing along the first dimension↙
(rows)

% Initialize outer product
outer_product2 = zeros(size(2, 2));  % Initialize to the correct size

for i=1:N_1000
```

```
    outer_product2  = outer_product2 + (X_1000(i,:)' - mean_estimator_1000')* ↙
(transpose(X_1000(i,:)' - mean_estimator_1000'));
end

varience_estimator_1000 = outer_product2/N_1000;
clear; clc; close all;

%% Question 2 %% Bayesian Parameter Estimation
rng('default') % For reproducibility

sigma   = 0.7;
% i)
mu1     = 3;
N_25    = 25;
x_i     = normrnd(mu1,sqrt(sigma),[N_25,1]);


% define values for y-axis
y       = zeros(length(x_i),1);
group   = ones(length(x_i), 1);  % All samples in one group
% Plot
figure;
gscatter(x_i, y, group, 'br', '.',18);
title('Scatter Plot of Normal Distribution for 25 samples');
xlabel("x")

% Maximum Likelihood Estimation
mean_estimator_i = sum(x_i)/N_25;

%ii)
mu_mu       = 2.8; % mean parameter of random variable mean
sigma_mu    = .8; % variance parameter of random variable mean

w1_ii       = (N_25*sigma_mu )/(N_25*sigma_mu  + sigma );
w2_ii       = (sigma )/(N_25*sigma_mu  + sigma );

mu_map_ii   = w1_ii * mean_estimator_i + w2_ii * mu_mu;

% iii)

N_1000  = 1000;
x_iii   = normrnd(mu1,sigma,[N_1000,1]);

% define values for y-axis
y       = zeros(length(x_iii),1);
group   = ones(length(x_iii), 1);  % All samples in one group
% Plot
figure;
gscatter(x_iii, y, group, 'br', '.',18);
title('Scatter Plot of Normal Distribution for 1000 samples');
xlabel("x")

% Maximum Likelihood Estimation
```

```
mean_estimator_iii = sum(x_iii)/N_1000;

% MAP Estimation
w1_iii          = (N_1000*sigma_mu )/(N_1000*sigma_mu  + sigma );
w2_iii          = (sigma )/(N_1000*sigma_mu  + sigma );

mu_map_iii      = w1_iii*mean_estimator_iii + w2_iii * mu_mu; % Prior infoya daha↙
yakın yorum yazarken aklında bulundurursun kıps :D


%% Question 3 %% Minimum error-rate classifier
rng(55)  % For reproducibility

% Mean vectors of classes
mu1     = [-1, -1]';
mu2     = [1, 1]';

% Covarience matrix
sigma   = [1.4 .2; .2 .28];

% Generate sets
omega1  = mvnrnd(mu1,sigma,500);
omega2  = mvnrnd(mu2,sigma,500);


% Randomly select 250 indices for training
indices1     = randperm(500, 250); % Randomly select 250 indices from 1 to 500
indices2     = randperm(500, 250); % For omega2 as well

% Split train-test sets using the random indices
omega1_train = omega1(indices1, :);
omega1_test  = omega1(setdiff(1:500, indices1), :); % Remaining points for testing
omega2_train = omega2(indices2, :);
omega2_test  = omega2(setdiff(1:500, indices2), :);

% Create a new figure for the plot
figure;

% Plot training data
scatter(omega1_train(:,1), omega1_train(:,2), 25, 'r', 'filled'); % Class 1↙
training data
hold on;
scatter(omega2_train(:,1), omega2_train(:,2), 25, 'b', 'filled'); % Class 2↙
training data
hold off;
title('Training Data (Class 1 and Class 2)');
xlabel('x1');
ylabel('x2');
legend('Class 1 Training', 'Class 2 Training');
grid on;

% Plot test data
figure;
```

```
scatter(omega1_test(:,1), omega1_test(:,2), 25, 'r', 'o'); % Class 1 test data
hold on
scatter(omega2_test(:,1), omega2_test(:,2), 25, 'b', 'o'); % Class 2 test data
hold off

title('Test Data');
xlabel('x1');
ylabel('x2');
legend( 'Class 1 Test', 'Class 2 Test');
grid on;


% Decision boundary from eq REFF
n   = (mu1 - mu2)'; % normal of the boundary line
x0  = (mu1 + mu2)/2;

% Decision boundary function
decision_boundary  = @(x1, x2) n * inv(sigma) * ([x1; x2] - x0);

% Classification and Error Calculation for Train Data
train_data           = [omega1_train; omega2_train];
train_labels         = [ones(250, 1); 2 * ones(250, 1)]; % 1 for class 1, 2 for↙
class 2
predicted_labels1    = zeros(size(train_labels));

for i = 1:length(train_data)
    x = train_data(i, :)';
    if decision_boundary(x(1), x(2)) > 0
        predicted_labels1(i) = 1; % Assign to class 1
    else
        predicted_labels1(i) = 2; % Assign to class 2
    end
end

% Calculate error rate
error_rate1 = sum(predicted_labels1 ~= train_labels) / length(train_labels);
fprintf('Error Rate for Train Data: %.2f%%\n', error_rate1 * 100);

% Classification and Error Calculation for Test Data
test_data           = [omega1_test; omega2_test];
test_labels         = [ones(250, 1); 2 * ones(250, 1)]; % 1 for class 1, 2 for↙
class 2
predicted_labels    = zeros(size(test_labels));

for i = 1:length(test_data)
    x = test_data(i, :)';
    if decision_boundary(x(1), x(2)) > 0
        predicted_labels(i) = 1; % Assign to class 1
    else
        predicted_labels(i) = 2; % Assign to class 2
    end
end
```

```
% Calculate error rate
error_rate = sum(predicted_labels ~= test_labels) / length(test_labels);
fprintf('Error Rate for Test Data: %.2f%%\n', error_rate * 100);

% Plot decision boundary on the train data
figure;
scatter(omega1_train(:, 1), omega1_train(:, 2), 25, 'r', 'filled'); % Class 1 test data
hold on;
scatter(omega2_train(:, 1), omega2_train(:, 2), 25, 'b', 'filled'); % Class 2 test data
fimplicit(@(x1, x2) decision_boundary(x1, x2), [-4 4 -4 4], 'k', 'LineWidth', 1.5);
hold off;

title(sprintf('Train Data with Decision Boundary (Error Rate: %.2f%%)', error_rate1 * 100));
xlabel('x1');
ylabel('x2');
legend('Class 1 Train', 'Class 2 Train', 'Decision Boundary');
grid on;

% Plot decision boundary on the test data
figure;
scatter(omega1_test(:, 1), omega1_test(:, 2), 25, 'r', 'o'); % Class 1 test data
hold on;
scatter(omega2_test(:, 1), omega2_test(:, 2), 25, 'b', 'o'); % Class 2 test data
fimplicit(@(x1, x2) decision_boundary(x1, x2), [-4 4 -4 4], 'k', 'LineWidth', 1.5);
hold off;

title(sprintf('Test Data with Decision Boundary (Error Rate: %.2f%%)', error_rate * 100));
xlabel('x1');
ylabel('x2');
legend('Class 1 Test', 'Class 2 Test', 'Decision Boundary');
grid on;

%% Question 3 iii)

% Maximum Likelihood estimator
N1              = size(omega1_train,1);
N2              = size(omega2_train,1);

mu1_estimator   = sum(omega1_train,1)/N1;
mu2_estimator   = sum(omega2_train,1)/N2;

% Initialize outer product

outer_product1 = zeros(size(2, 2));  % Initialize to the correct size

for i=1:N1
    outer_product1 = outer_product1 + (omega1_train(i,:)' - mu1_estimator')*(transpose(omega1_train(i,:)' - mu1_estimator'));
end
```

```
variance_estimator1 = outer_product1/N1;

outer_product2 = zeros(size(2, 2));  % Initialize to the correct size

for i=1:N2
    outer_product2  = outer_product2 + (omega2_train(i,:)' - mu2_estimator')*↙
(transpose(omega2_train(i,:)' - mu2_estimator'));
end

variance_estimator2 = outer_product2/N2;

% Decision boundary from eq REFF
n_2   = (mu1_estimator - mu2_estimator); % normal of the boundary line
x0_2  = (mu1_estimator + mu2_estimator)'/2;

% Decision boundary function
decision_boundary_2  = @(x1_2, x2_2) n_2 * inv(variance_estimator1) * ([x1_2; x2_2]↙
- x0_2);

% Classification and Error Calculation for Train Data
train_data_2           = [omega1_train; omega2_train];
train_labels_2         = [ones(250, 1); 2 * ones(250, 1)]; % 1 for class 1, 2 for↙
class 2
predicted_labels1_2     = zeros(size(train_labels));

for i = 1:length(train_data_2)
    x = train_data_2(i, :)';
    if decision_boundary_2(x(1), x(2)) > 0
        predicted_labels1_2(i) = 1; % Assign to class 1
    else
        predicted_labels1_2(i) = 2; % Assign to class 2
    end
end

% Calculate error rate
error_rate1_2 = sum(predicted_labels1_2 ~= train_labels_2) / length↙
(train_labels_2);
fprintf('Error Rate for Train Data ML: %.2f%%\n', error_rate1_2 * 100);

% Classification and Error Calculation for Test Data
test_data_2           = [omega1_test; omega2_test];
test_labels_2         = [ones(250, 1); 2 * ones(250, 1)]; % 1 for class 1, 2 for↙
class 2
predicted_labels_2    = zeros(size(test_labels));

for i = 1:length(test_data_2)
    x = test_data_2(i, :)';
    if decision_boundary_2(x(1), x(2)) > 0
        predicted_labels_2(i) = 1; % Assign to class 1
    else
        predicted_labels_2(i) = 2; % Assign to class 2
    end
```

```
end

% Calculate error rate
error_rate_2 = sum(predicted_labels_2 ~= test_labels_2) / length(test_labels_2);
fprintf('Error Rate for Test Data ML: %.2f%%\n', error_rate_2 * 100);

% Plot decision boundary on the train data
figure;
scatter(omega1_train(:, 1), omega1_train(:, 2), 25, 'r', 'filled'); % Class 1 test data
hold on;
scatter(omega2_train(:, 1), omega2_train(:, 2), 25, 'b', 'filled'); % Class 2 test data
fimplicit(@(x1_2, x2_2) decision_boundary_2(x1_2, x2_2), [-4 4 -4 4], 'k', 'LineWidth', 1.5);
hold off;

title(sprintf('Train Data with Decision Boundary ML (Error Rate: %.2f%%)', error_rate1_2 * 100));
xlabel('x1');
ylabel('x2');
legend('Class 1 Train', 'Class 2 Train', 'Decision Boundary');
grid on;

% Plot decision boundary on the test data
figure;
scatter(omega1_test(:, 1), omega1_test(:, 2), 25, 'r', 'o'); % Class 1 test data
hold on;
scatter(omega2_test(:, 1), omega2_test(:, 2), 25, 'b', 'o'); % Class 2 test data
fimplicit(@(x1_2, x2_2) decision_boundary_2(x1_2, x2_2), [-4 4 -4 4], 'k', 'LineWidth', 1.5);
hold off;

title(sprintf('Test Data with Decision Boundary ML (Error Rate: %.2f%%)', error_rate_2 * 100));
xlabel('x1');
ylabel('x2');
legend('Class 1 Test', 'Class 2 Test', 'Decision Boundary');
grid on;

%% Question 4 %%  Non-parametric Density Estimation
% Parameters of normal distributed feature vector
mu          = [-1.5; 1.5];
sigma       = [.8 .2; .2 .6];

% Number of samples
N           = 50;  % Reduced sample size for faster computation

% Grid for plotting the Gaussian distribution
[X1, X2]    = meshgrid(linspace(-6, 6, 500), linspace(-6, 6, 500));
X           = [X1(:) X2(:)];

% True distribution
```

```
gaus_true   = mvnpdf([X1(:) X2(:)], mu', sigma);
gaus_true   = reshape(gaus_true, length(X2), length(X1));

% Plot true distribution
figure;
surf(X1, X2, gaus_true, 'EdgeColor', 'interp');
title('True Distribution')
xlabel('X1')
ylabel('X2')
zlabel('PDF')

% Generate samples from the distribution
X_samples = mvnrnd(mu, sigma, N);

% Different h1 values for testing
h1_values = [0.5, 1, 2, 3, 5, 10];
num_h1 = length(h1_values);

% Loop over each h1 value and perform Parzen window density estimation

% Gaussian Parzen window estimation
figure;
for idx = 1:num_h1
    h1  = h1_values(idx);
    % Initial volume
    Vo  = h1^3;
    % Adaptive volume and h
    V   = Vo / sqrt(N);
    h   = V^(1/3);

    % Estimate density with Gaussian Parzen window
    p_gaussian = zeros(size(X,1), 1);   % Initialize density function
    for k = 1:size(X,1)
        % Take one feature from true distribution
        x   = X(k, :);

        sum = 0;
        for n = 1:N
            % Calculate each samples effect
            xk  = X_samples(n, :);
            sum = sum + parzen_window((x - xk) / h);

        end
        p_gaussian(k) = sum / (N * V);
    end
    p_gaussian = reshape(p_gaussian, length(X2), length(X1));

    % Plot the Gaussian Parzen window estimate for the current h1
    subplot(2, 3, idx);  % Arrange in a 2x3 grid
    surf(X1, X2, p_gaussian, 'EdgeColor', 'interp');
    title(['Gaussian Parzen (h1 = ', num2str(h1), ')(V = ', num2str(V), ')']);
    xlabel('X1');
    ylabel('X2');
```

```
    zlabel('PDF');
end

% Cubic Parzen window estimation
figure;
for idx = 1:num_h1
    h1 = h1_values(idx);
    Vo = h1^3;
    V = Vo / sqrt(N);
    h = V^(1/3);

    % Estimate density with cubic Parzen window
    p_cubic = zeros(size(X,1), 1);
    for k = 1:size(X,1)
        x = X(k, :);
        sum = 0;
        for n = 1:N
            xk = X_samples(n, :);
            sum = sum + parzen_window_cub((x - xk) / h);
        end
        p_cubic(k) = sum / (N * V);
    end
    p_cubic = reshape(p_cubic, length(X2), length(X1));

    % Plot the Cubic Parzen window estimate for the current h1
    subplot(2, 3, idx);  % Arrange in a 2x3 grid
    surf(X1, X2, p_cubic, 'EdgeColor', 'interp');
    title(['Cubic Parzen (h1 = ', num2str(h1), ')(V = ', num2str(V), ')']);
    xlabel('X1');
    ylabel('X2');
    zlabel('PDF');
end

%% Question 5
% Load dataset
load fisheriris

% Extract feature and labels
X = meas(:,1:2);     % Features, take length and width
Y = species;         % Labels

figure;
gscatter(X(:,1),X(:,2),species,'rgb','osd');
xlabel('Sepal length');
ylabel('Sepal width');
% PCA
[coeff, score, eigenvalues] = pca(X);

% Take basis and normal vectors
basis   = coeff(:,1);
normal  = coeff(:,2);

% Mean vector of data
```

```
m       = mean(X,1);

% Scores give the projection of each point onto the line
[n,p]   = size(X);

% Fitting lines
Xfit    = repmat(m,n,1) + score(:,1)*coeff(:,1)'; % Most principle
Xfit2   = repmat(m,n,1) + score(:,2)*coeff(:,2)'; % Least principle


% Error
error   = abs((X - repmat(m,n,1))*normal);
sse     = sum(error.^2);

% Plot PCA
figure;
t = [min(score(:,1)) - 0.2, max(score(:,1)) + 0.2];
endpts = [m + t(1) * basis'; m + t(2) * basis'];
plot(endpts(:,1), endpts(:,2), 'k-', 'LineWidth', 1.5);
hold on;
colors = 'rgb';
classes = unique(Y);

idx1 = strcmp(Y, classes{1});
idx2 = strcmp(Y, classes{2});
idx3 = strcmp(Y, classes{3});

plot(X(idx1,1), X(idx1,2), 'ro', 'MarkerFaceColor', 'none');  % Blue diamond↙
outline with no fill
plot(X(idx2,1), X(idx2,2), 'gs', 'MarkerFaceColor', 'none');
plot(X(idx3,1), X(idx3,2), 'bd', 'MarkerFaceColor', 'none');


for i = 1:length(classes)
    % Select data points for the current class
    idx = strcmp(Y, classes{i});
    % Plot projections on the PCA line
    X1 = [X(idx,1) Xfit(idx,1) nan*ones(sum(idx),1)];
    X2 = [X(idx,2) Xfit(idx,2) nan*ones(sum(idx),1)];
    plot(X1', X2', '-', 'Color', colors(i));
end

hold off;
legend('Fitting Line','Setosa','versicolor','virginica', 'Location', 'best');
xlabel('Sepal length');
ylabel('Sepal width');
title('Orthogonal Regression using PCA with Projections by Class');


% Plot of most principle direction
X_setosa        = Xfit(1:50,1);
X_versicolor    = Xfit(50:100,1);
X_virginica     = Xfit(100:150,1);
```

```
figure;
subplot(1,2,1)
histogram(X_setosa, 10); % 10 bins for Setosa
hold on;
histogram(X_versicolor, 10); % 10 bins for Versicolor
histogram(X_virginica, 10); % 10 bins for Virginica
hold off;
title("Histogram of of the PCA-reduced 1D data on most principle direction")
xlabel('Principal Component 1');
ylabel('Frequency');

% Plot of least principle direction
X_setosa2        = Xfit2(1:50,1);
X_versicolor2    = Xfit2(50:100,1);
X_virginica2     = Xfit2(100:150,1);

subplot(1,2,2)
histogram(X_setosa2, 10); % 10 bins for Setosa
hold on;
histogram(X_versicolor2, 10); % 10 bins for Versicolor
histogram(X_virginica2, 10); % 10 bins for Virginica
hold off;
title("Histogram of of the PCA-reduced 1D data on least principle direction")
xlabel('Principal Component 2');
ylabel('Frequency');
%% Parzen Functions
% Gaussian shape Parzen window function
function f = parzen_window(u)
    u = norm(u);
    f = (1/sqrt(2 * pi)) * exp(-0.5 * u^2);
end

% Cubic Parzen window function
function f_cub = parzen_window_cub(u)
    if norm(u) <= 1/2
        f_cub = 1;
    else
        f_cub = 0;
    end
end
```