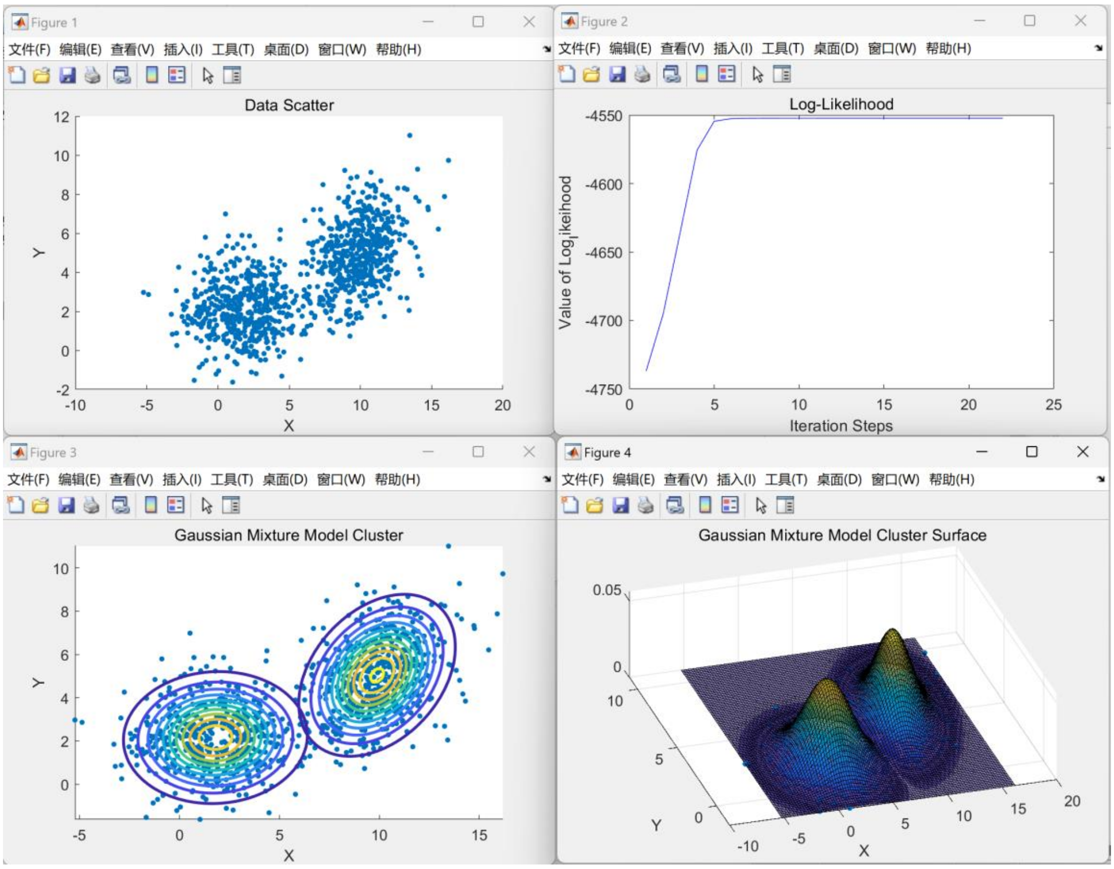
# ENGN 2520 Pattern Recognition and Machine Learning

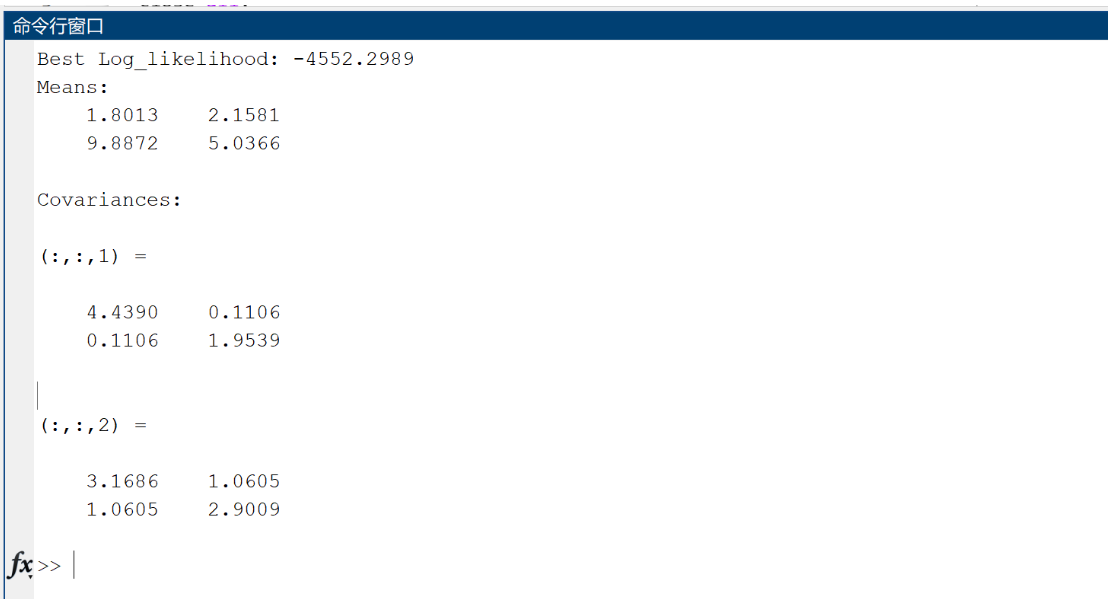
# Homework 5

Zhuo Wang

## Problem 1

(1) A mixture of 2 Gaussians





```
Best Log_likelihood: -4552.2989
Means:
    1.8013    2.1581
    9.8872    5.0366

Covariances:

(:,:,1) =

    4.4390    0.1106
    0.1106    1.9539


(:,:,2) =

    3.1686    1.0605
    1.0605    2.9009

fx >>
```

## (2) A mixture of 3 Gaussians

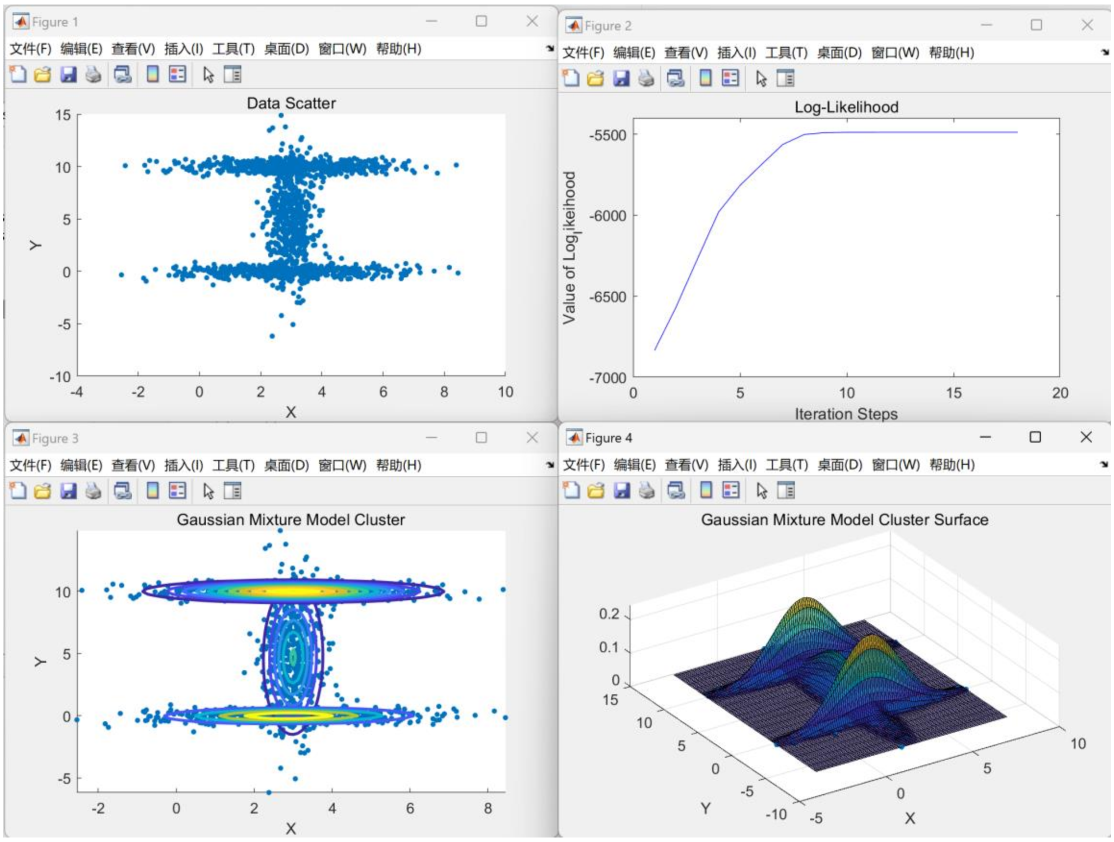
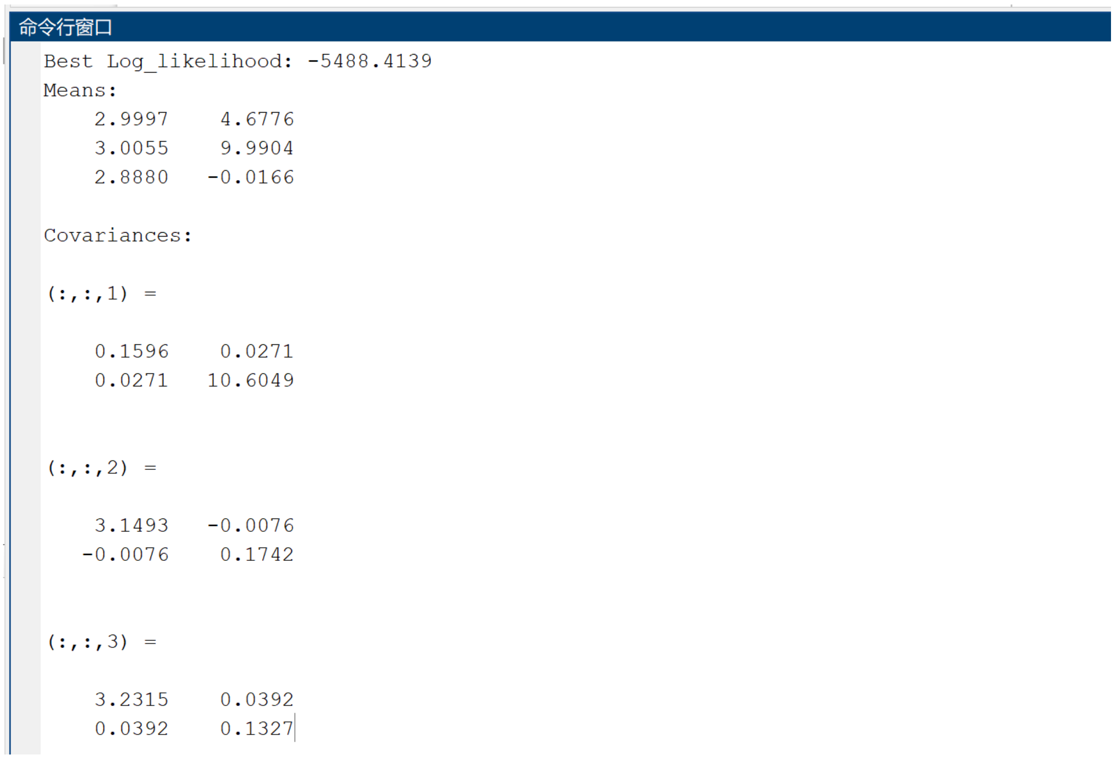


Figure 1
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Data Scatter
15
10
5
0
-5
-10
Y
-4
-2
0
2
4
6
8
10
X
Figure 2
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Log-Likelihood
-5500
-6000
-6500
-7000
Value of Log_likeihood
0
5
10
15
20
Iteration Steps
Figure 3
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Gaussian Mixture Model Cluster
10
5
0
-5
Y
-2
0
2
4
6
8
X
Figure 4
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Gaussian Mixture Model Cluster Surface
0.2
0.1
0
15
10
5
0
-5
-10
Y
-5
0
5
10
X



```
Best Log_likelihood: -5488.4139
Means:
    2.9997    4.6776
    3.0055    9.9904
    2.8880   -0.0166

Covariances:

(:,:,1) =

    0.1596    0.0271
    0.0271   10.6049


(:,:,2) =

    3.1493   -0.0076
   -0.0076    0.1742


(:,:,3) =

    3.2315    0.0392
    0.0392    0.1327
```

# Source Code

GaussianMixtureModelClustering.m

```
clc;
clear;
close all;

data = load('data3.mat').data;

figure();
scatter(data(:,1),data(:,2),150,'.');
title('Data Scatter');
xlabel('X');
ylabel('Y');

in = K(data, 3);

[f, l] = M(data, in, 0.000001);

disp("Best Log_likelihood: " + l(end));
disp("Means:");
disp(f.mu);
disp("Covariances:");
disp(f.Sigma);

P(data, f);
```

K.m

```matlab
function initial_estimates = K(data, K)
    idx = randperm(size(data, 1), K);
    centroids = data(idx, :);

    initial_estimates = struct('mu', [], 'Sigma', [], 'pi', []);

    initial_estimates.mu = centroids;

    covariance = cov(data);

    covariance_multiplier = 1;

    initial_covariance = covariance * covariance_multiplier;

    initial_estimates.Sigma = repmat(initial_covariance, [1, 1, K]);

    initial_estimates.pi = ones(1, K) / K;
end
```

M.m

```
function [final_parameters, log_likelihoods] = M(X, initial_parameters, 
convergence_threshold)

    parameters_old = initial_parameters;
    log_likelihoods = [];

    i = 1;
    is = [];

    while true
        is = [is, i];
        responsibilities = compute_responsibilities(X, parameters_old);

        [mu_new, Sigma_new, pi_new] = update_parameters(X, 
responsibilities);

        parameters_new = struct('mu', mu_new, 'Sigma', Sigma_new, 'pi', 
pi_new);

        log_likelihood = compute_log_likelihood(X, mu_new, Sigma_new, 
pi_new);
        log_likelihoods = [log_likelihoods, log_likelihood];

        i = i + 1;

        if should_stop(log_likelihoods, convergence_threshold)
            figure();
            plot(is, log_likelihoods, '-b');
            title('Log-Likelihood');
            xlabel('Iteration Steps');
            ylabel('Value of Log_likeihood');
            break;
        end

        parameters_old = parameters_new;
    end

    final_parameters = parameters_old;
end

function responsibilities = compute_responsibilities(X, parameters)
    mu = parameters.mu;
    Sigma = parameters.Sigma;
    pi = parameters.pi;

    N = size(X, 1);

    K = size(mu, 1);

    responsibilities = zeros(N, K);

    for n = 1:N
        for k = 1:K
            responsibilities(n, k) = pi(k) * mvnpdf(X(n, :), mu(k, :),
Sigma(:, :, k));
        end
        responsibilities(n, :) = responsibilities(n, :) /
sum(responsibilities(n, :));
    end
end

function [mu_new, Sigma_new, pi_new] = update_parameters(X,
responsibilities)
    [N, D] = size(X);
    K = size(responsibilities, 2);

    Nk = sum(responsibilities, 1);

    mu_new = zeros(K, D);
    for k = 1:K
        mu_new(k, :) = sum(repmat(responsibilities(:, k), 1, D) .* X, 1) /
Nk(k);
    end

    Sigma_new = zeros(D, D, K);
    for k = 1:K
        diff = bsxfun(@minus, X, mu_new(k, :));
        Sigma_new(:, :, k) = (repmat(responsibilities(:, k), 1, D) .* diff)'
* diff / Nk(k);
    end

    pi_new = Nk / N;
end

function log_likelihood = compute_log_likelihood(X, mu, Sigma, pi)
    [N, ~] = size(X);
    K = size(mu, 1);
    log_likelihood = 0;

    for n = 1:N
        sum_prob = 0;
        for k = 1:K
            prob = pi(k) * mvnpdf(X(n, :), mu(k, :), Sigma(:, :, k));
            sum_prob = sum_prob + prob;
        end
        log_likelihood = log_likelihood + log(sum_prob);
    end
end

function stop = should_stop(log_likelihoods, convergence_threshold)
    stop = false;

    if numel(log_likelihoods) > 1
        delta_ll = abs(log_likelihoods(end) - log_likelihoods(end-1));
        if delta_ll < convergence_threshold
            stop = true;
        end
    end
end
```

P.m

```
function P(data, gm_struct)
    figure();
    scatter(data(:,1),data(:,2),150,'.');
    hold on;

    mu = gm_struct.mu;
    Sigma = gm_struct.Sigma;
    pi = gm_struct.pi;

    K = size(mu, 1);

    for k = 1:K
        x1 = linspace(min(data(:,1)), max(data(:,1)), 100);
        x2 = linspace(min(data(:,2)), max(data(:,2)), 100);
        [X1, X2] = meshgrid(x1, x2);
        X = [X1(:) X2(:)];

        pdf_values = mvnpdf(X, mu(k,:), Sigma(:,:,k));
        pdf_matrix = reshape(pdf_values, length(x2), length(x1));

        contour(X1, X2, pdf_matrix, 'LineWidth', 2);
    end

    title('Gaussian Mixture Model Cluster');
    xlabel('X');
    ylabel('Y');
    hold off;

    figure();
    scatter3(data(:,1), data(:,2), zeros(size(data, 1), 1), 150, '.');
    hold on;

    for k = 1:K
        x1 = linspace(min(data(:,1)), max(data(:,1)), 100);
        x2 = linspace(min(data(:,2)), max(data(:,2)), 100);
        [X1, X2] = meshgrid(x1, x2);
        X = [X1(:) X2(:)];

        pdf_values = mvnpdf(X, mu(k,:), Sigma(:,:,k));
        pdf_matrix = reshape(pdf_values, length(x2), length(x1));

        surf(X1, X2, pdf_matrix, 'FaceAlpha', 0.5);
    end

    title('Gaussian Mixture Model Cluster Surface');
    xlabel('X');
    ylabel('Y');
    hold off;
end
```