ECE 271A: Statistical Learning I - Quiz #5

Mazeyu Ji - A59023027

December 1, 2023

1. Basic principle

The EM update equations for the parameters of a Gaussian Mixture Model are as follows:

- **E-step:**

$$h_{ij} = \frac{G(x_i, \mu_j^{(n)}, \sigma_j^{(n)})\pi_j^{(n)}}{\sum_{k=1}^{C} G(x_i, \mu_k^{(n)}, \sigma_k^{(n)})\pi_k^{(n)}}$$

- **M-step:**

$$\mu_j^{(n+1)} = \frac{\sum_i h_{ij} x_i}{\sum_i h_{ij}}$$

$$\pi_j^{(n+1)} = \frac{1}{n}\sum_i h_{ij}$$

$$\sigma_j^{2(n+1)} = \frac{\sum_i h_{ij}(x_i - \mu_j)^2}{\sum_i h_{ij}}$$

2. Results Analysis

2.1 Problem (a)

The relationship between the probability of error and the number of dimensions for each of the 25 classifiers reveals that due to random initialization, each classifier converges to different local optima. This means that even for the same problem, the final model parameters can vary because of different starting points. This randomness in initialization leads to performance variations across different classifiers, even when the same training data and algorithm are used.

As the number of dimensions increases, the differences between classifiers become more pronounced. This could be because in higher-dimensional spaces, the impact of different feature combinations and model initialization methods on the probability of error becomes more complex and varied. However, the overall trend of error rate versus the number of dimensions is similar across different classifiers, indicating that despite the specific performance of each being different, their performance across dimensions exhibits certain consistencies. This is likely because certain dimensions provide more useful information for classification, and thus, across these dimensions, different classifiers show a reduction in error rates.

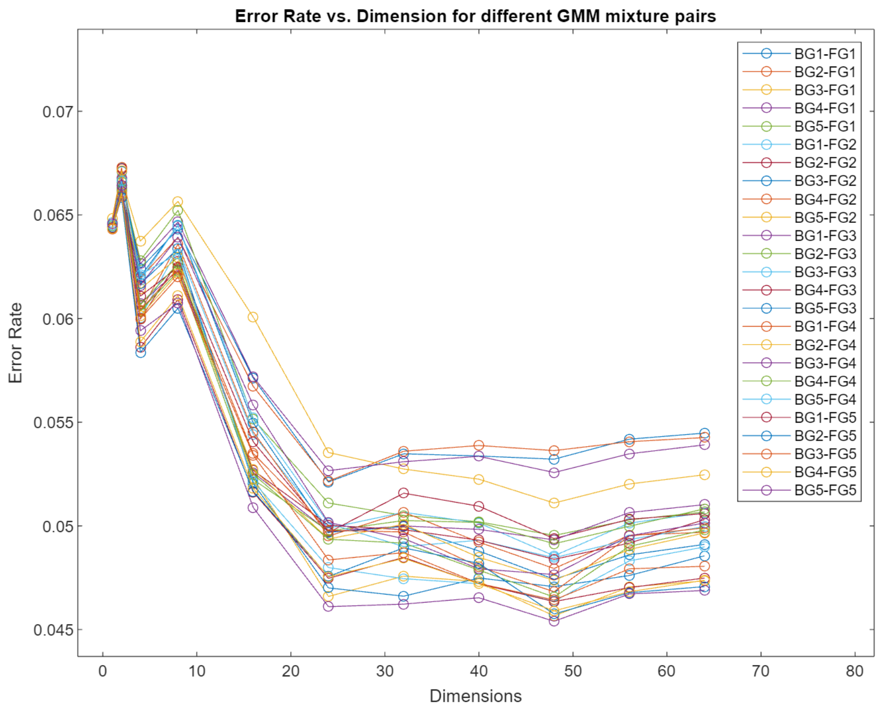

Figure 1: Error Rate vs. Dimension for different GMM mixture pairs.

2.2 Problem (b)

In analyzing the impact of the number of Gaussian mixture components C on the classification error rate, it was observed that:

- With $C = 1$, the classifier exhibits the highest error rate, which may even increase with the dimension, indicating a poor capture of the data complexity.
- For $C = 2$, while the error rate is lower than that with a single component, the error rate initially decreases as the dimension increases, suggesting an improvement in capturing the data's structure. However, beyond a certain dimension, the error rate begins to rise, indicating a possible overfitting or an increased sensitivity to noise in the higher-dimensional space.
- When C reaches 4, the error rate decreases and stabilizes, suggesting that adding more components improves performance to an extent, but beyond a certain threshold, further additions offer diminishing returns.

This highlights the importance of selecting an appropriate number of components while accounting for the dimensionality's influence in Gaussian mixture models for classification.

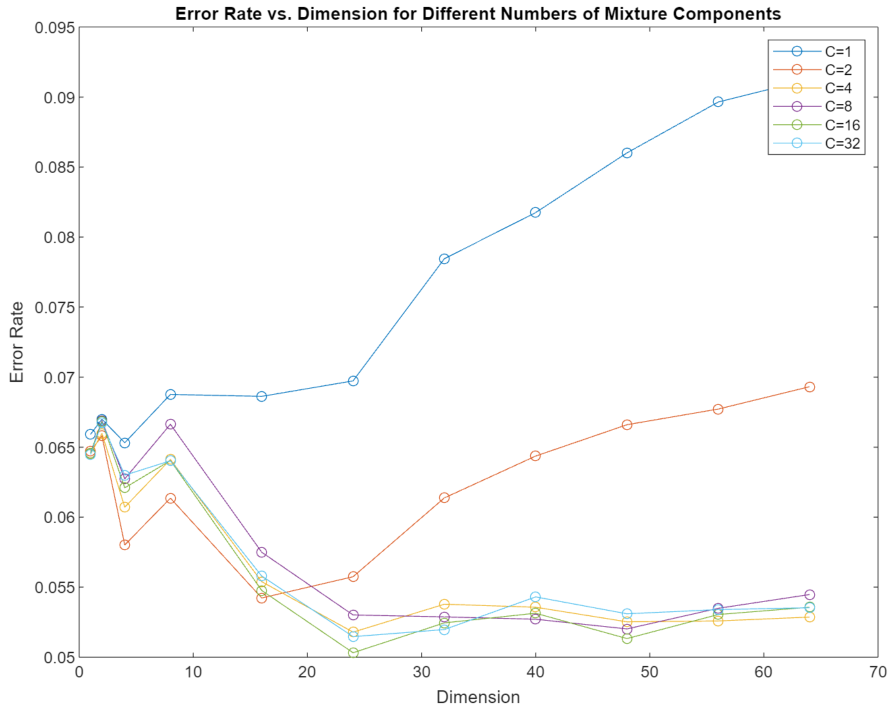

Figure 2: Error Rate vs. Dimension for Different Numbers of Mixture Components.

Appendix

(a)solution.m

```
%% Load the data
load('TrainingSamplesDCT_8_new.mat'); % Assume
    TrainingSamplesDCT_new_8.mat data is loaded into the
    workspace

%% Train EM models
% Train a set of five GMMs with C=8 components
for i = 1:5
    % Train and save GMM parameters for the i-th model
    trainEM(8, TrainsampleDCT_BG, TrainsampleDCT_FG, i); %
        Labels 1-5
end
```

```matlab
% Train a set of GMMs for different values of C components
C_values = [1, 2, 4, 8, 16, 32];
for C = C_values
    % Train and save GMM parameters for each C value, labeled 0
    trainEM(C, TrainsampleDCT_BG, TrainsampleDCT_FG, 0);
end

%% (a) 25 random classifiers
C = 8; % Number of components in the mixture model
dimensions = [1, 2, 4, 8, 16, 24, 32, 40, 48, 56, 64]; % Array of dimensions to be considered
numMixtures = 5; % Number of GMMs to be trained for each class
errorRates = zeros(numMixtures^2, length(dimensions)); % Matrix to store error rates
mixturePairs = combvec(1:numMixtures, 1:numMixtures)'; % Generate all possible mixture pairs

% Calculate error rates for each dimension
for d = 1:length(dimensions)
    dimension = dimensions(d);
    disp(['Starting dimension: ', num2str(dimension)]);

    % Loop through all possible pairs of mixtures
    for mixPair = 1:size(mixturePairs, 1)
        mixBG = mixturePairs(mixPair, 1); % Background mixture index
        mixFG = mixturePairs(mixPair, 2); % Foreground mixture index

        % Load GMM parameters for background and foreground
        load(sprintf('trainedGMM_Comp=%d_Label=%d.mat', C, mixBG), 'weightsBG', 'meansBG', 'covariancesBG');
        load(sprintf('trainedGMM_Comp=%d_Label=%d.mat', C, mixFG), 'weightsFG', 'meansFG', 'covariancesFG');

        % Compute error rates using the EM_BDR function
        error = EM_BDR(TrainsampleDCT_BG, TrainsampleDCT_FG, weightsBG, weightsFG, meansBG(:, 1:dimension), meansFG(:, 1:dimension), covariancesBG(1:dimension, 1:dimension, :), covariancesFG(1:dimension, 1:dimension, :), dimension, C);

        % Store the error rate
        errorRates(mixPair, d) = error;
    end
end

%% Plot the error rates
```

```matlab
figure;
for mixPair = 1:size(mixturePairs, 1)
    plot(dimensions, errorRates(mixPair, :), '-o'); % Plot a line for each mixture pair
    hold on; % Hold the figure for multiple plots
end
hold off;
xlabel('Dimensions');
ylabel('Error Rate');
title('Error Rate vs. Dimension for different GMM mixture pairs');
legendCell = cellstr(num2str(mixturePairs, 'BG%d-FG%d')); % Create a legend
legend(legendCell);

%% (b) Classifiers with different component numbers
% Initialize parameters
dimensions = [1, 2, 4, 8, 16, 24, 32, 40, 48, 56, 64]; % Array of dimensions to be considered
C_values = [1, 2, 4, 8, 16, 32]; % Array of numbers of mixture components
errorRatesC = zeros(length(C_values), length(dimensions)); % Matrix to store error rates for different C values

% Loop to calculate error rates for each value of C
for idx = 1:length(C_values)
    C = C_values(idx); % Current number of mixture components
    disp(['Starting component num: ', num2str(C)]);

    % Loop through different dimensions
    for d = 1:length(dimensions)
        dimension = dimensions(d);

        % Load the model parameters, assuming models are saved with iteration label 0
        load(sprintf('trainedGMM_Comp=%d_Label=0.mat', C), 'weightsBG', 'meansBG', 'covariancesBG');
        load(sprintf('trainedGMM_Comp=%d_Label=0.mat', C), 'weightsFG', 'meansFG', 'covariancesFG');

        % Compute and store the error rate
        errorRatesC(idx, d) = EM_BDR(TrainsampleDCT_BG, TrainsampleDCT_FG, weightsBG, weightsFG, meansBG(:, 1:dimension), meansFG(:, 1:dimension), covariancesBG(1:dimension, 1:dimension, :), covariancesFG(1:dimension, 1:dimension, :), dimension, C);
    end
end
```

```matlab

%% Plot the error rates
figure;
for idx = 1:length(C_values)
    plot(dimensions, errorRatesC(idx, :), '-o', 'DisplayName', sprintf('C=%d', C_values(idx)));
    hold on; % Hold on to plot multiple lines on the same figure
end
hold off; % Release the hold to finish plotting
xlabel('Dimension');
ylabel('Error Rate');
title('Error Rate vs. Dimension for Different Numbers of Mixture Components');
legend show;
```

(b)trainEM.m

```matlab
function [] = trainEM(numComponents, dataBG, dataFG, iterationLabel)
    numIterations = 2000; % Number of iterations for the EM algorithm
    numFeatures = size(dataBG, 2); % Assuming the number of features is the number of columns in dataBG

    % Initialize parameters for background and foreground
    [weightsBG, meansBG, covariancesBG] = initializeGMMParameters(numComponents, dataBG);
    [weightsFG, meansFG, covariancesFG] = initializeGMMParameters(numComponents, dataFG);

    % Train the GMM for background data
    fprintf('Starting EM for BG with %d components, iteration %d\n', numComponents, iterationLabel);
    [weightsBG, meansBG, covariancesBG] = runEM(dataBG, numComponents, numIterations, weightsBG, meansBG, covariancesBG);

    % Train the GMM for foreground data
    fprintf('Starting EM for FG with %d components, iteration %d\n', numComponents, iterationLabel);
    [weightsFG, meansFG, covariancesFG] = runEM(dataFG, numComponents, numIterations, weightsFG, meansFG, covariancesFG);

    % Save the trained model parameters
    saveFileName = sprintf('trainedGMM_Comp=%d_Label=%d.mat', numComponents, iterationLabel);
```

```
    save(saveFileName, 'weightsBG', 'meansBG', 'covariancesBG', 'weightsFG', 'meansFG', 'covariancesFG');
end

function [weights, means, covariances] = initializeGMMParameters(numComponents, data)
    % Initialize parameters for the GMM
    [n, d] = size(data);
    weights = rand (numComponents ,1) ;
    weights = weights / sum (weights) ;
    means = data(randperm(n, numComponents), :);
    covariances = repmat(diag(diag(rand(d))), [1, 1, numComponents]);
end

function [weights, means, covariances] = runEM(data, numComponents, numIterations, weights, means, covariances)
    % Run the EM algorithm for GMM
    n = size(data, 1);
    numFeatures = size(data, 2);

    for iteration = 1:numIterations
        % E-step: compute responsibilities
        responsibilities = zeros(n, numComponents);
        for j = 1:numComponents
            % Calculate the covariance matrix for the j-th component
            covarianceMatrix = reshape(covariances(:, :, j), numFeatures, numFeatures);
            % Compute the probability density function for each data point
            responsibilities(:, j) = weights(j) * mvnpdf(data, means(j, :), covarianceMatrix);
        end
        responsibilities = responsibilities ./ sum(responsibilities, 2);

        % M-step: update weights, means, and covariances
        for j = 1:numComponents
            weight = sum(responsibilities(:, j)) / n;
            mean = (responsibilities(:, j)' * data) / sum(responsibilities(:, j));
            centeredData = data - mean;
            covariance = (centeredData' * (centeredData .* responsibilities(:, j))) / sum(responsibilities(:, j));

            weights(j) = weight;
```

```
                means(j, :) = mean;
                covariances(:, :, j) = diag(diag(covariance));
            end

    end
end
```

(c)EM_BDR.m

```
function [error] = EM_BDR(trainBG, trainFG, weightBG,
    weightFG, muBG, muFG, sigmaBG, sigmaFG, dimension,
    numComponents)
    % Estimate the prior probabilities
    [rowFG, ~] = size(trainFG);
    [rowBG, ~] = size(trainBG);
    priorFG = rowFG / (rowBG + rowFG);
    priorBG = rowBG / (rowBG + rowFG);

    % Read and preprocess the image
    img = im2double(imread('cheetah.bmp'));
    [height, width] = size(img);

    % Get zig-zag pattern
    pattern = readmatrix('Zig-Zag Pattern.txt') + 1;

    % Initialize result mask
    maskResult = zeros(height-7, width-7);

    % Loop over the image
    for i = 1:(height - 7)
        for j = 1:(width - 7)
            block = img(i:(i+7), j:(j+7));
            dctBlock = dct2(block);
            zigzagBlock = zeros(1, 64);
            for m = 1:8
                for n = 1:8
                    zigzagBlock(pattern(m,n)) = dctBlock(m,n
                        );
                end
            end

            % Select the corresponding parameters for FG and
                BG
            muFgD = muFG(:, 1:dimension);
            muBgD = muBG(:, 1:dimension);
            sigmaFgD = sigmaFG(1:dimension, 1:dimension, 1:
                numComponents);
            sigmaBgD = sigmaBG(1:dimension, 1:dimension, 1:
                numComponents);
```

```
            % Compute the likelihoods for FG and BG
            pxYFg = myGmmPdf(zigzagBlock(1:dimension), muFgD, sigmaFgD, weightFG);
            pxYBg = myGmmPdf(zigzagBlock(1:dimension), muBgD, sigmaBgD, weightBG);

            % Apply BDR
            if pxYFg * priorFG > pxYBg * priorBG
                maskResult(i, j) = 1;
            end
        end
    end

    % % Display the masks
    % figure;
    % imshow(maskResult);
    % title('Classification Results');

    % Compute classification error against ground truth
    maskGT = im2double(imread('cheetah_mask.bmp'));
    error = sum(sum(maskResult ~= maskGT(1:end-7, 1:end-7))) / numel(maskResult);
end

% My own GMM function
function pdfValue = myGmmPdf(x, mu, sigma, weight)
    numComponents = size(mu, 1);
    pdfValue = 0;
    for i = 1:numComponents
        diff = x - mu(i,:);
        exponent = -0.5 * (diff / sigma(:,:,i)) * diff';
        coefficient = 1 / sqrt(((2 * pi)^length(x)) * det(sigma(:,:,i)));
        pdfValue = pdfValue + weight(i) * coefficient * exp(exponent);
    end
end
```