Homework Assignment 5

Quan Luo
Statistical Learning I - Fall 2021

December 4, 2021

Problem Quiz. This week we use the cheetah image to evaluate the performance of a classifier based on mixture models estimated with EM. Once again we use the decomposition into 8×8 image blocks, compute the DCT of each block, and zig-zag scan. For this (using the data in *TrainingSamplesDCT_new_8.mat*) we fit a mixture of Gaussians of diagonal covariance to each class, i.e.

$$P_{X|Y}(x|i) = \sum_{c=1}^{C} \pi_c G(x, \mu_c, \Sigma_c)$$

(a) For each class, learn 5 mixtures of $C = 8$ components, using a random initialization (recall that the mixture weights must add up to one). Plot the probability of error vs. dimension for each of the 25 classifiers obtained with all possible mixture pairs. Comment the dependence of the probability of error on the initialization.

Sol. The probability of error vs. dimension result is as follow. Note that the codes are attached at the last of the report.

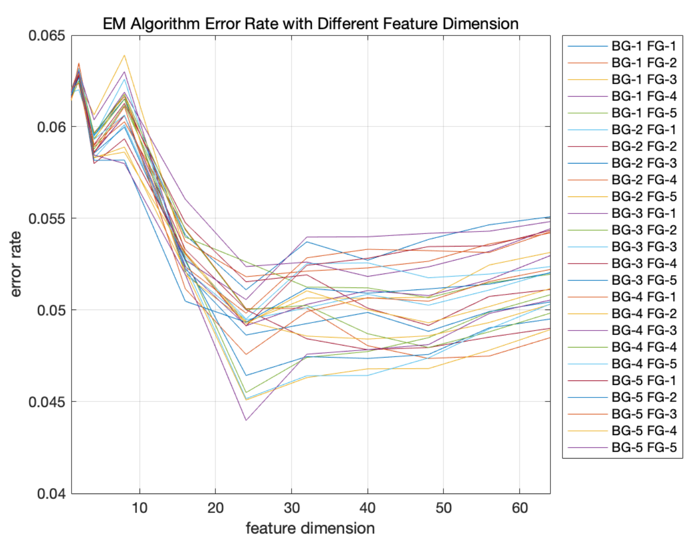

Figure 1: EM Algorithm Test with Different Feature Dimension

Comments:

(a) **Dependence of probability of error on the initialization:** Using different initialization (randomly initialized every time) is really important for EM algorithm. We can find that distinct initialization may result in higher or lower probability error. That's telling us that some of them are falling into the local optimum of the result. Some of them is smaller than the others and some of them are bigger while all of them are the same algorithm process. However, it's also important to see that the trend of the error with respect to dimension is not changing with different initialization. That's because it's determined by the feature dimension.

(b) **Comment on Feature Dimension:** We can see that when the feature dimension is getting higher, the probability of error may go up or down. We've shown in the previous homework, it's better when we choose the best features rather than directly use the 64 dimensional feature. Thus this is also an reasonable interpretation of the result we got before.

□

(b) For each class, learn mixtures with $C \in \{1, 2, 4, 8, 16, 32\}$. Plot the probability of error vs. dimension for each number of mixture components. What is the effect of the number of mixture components on the probability of error?

Sol. The probability of error vs. dimension result for each number of mixture components is as follow.

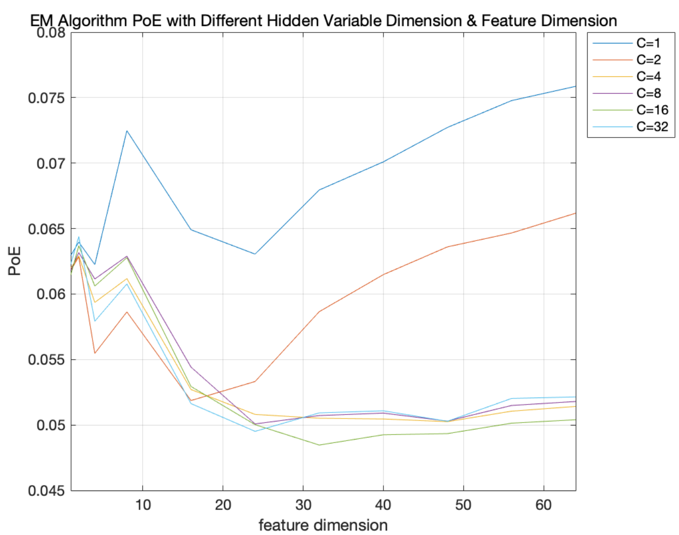

Figure 2: EM Algorithm Test with Different Feature Dimension for Different Hidden Dimension

When $C = 1$ or $C = 2$ the plot is really different from the others. That's reasonable because when C is small, it may lead to bias of estimation. Notice that when $C = 32$ it's worse than the result of $C = 16$. That's also reasonable because maybe there are overfitting by too many hidden dimension. For the others, the error decreases as C increases. Note that the initialization will also disturb the result. □

Codes are attached as follow.

```
% EM.m
load('/Users/quan/Documents/MATLAB/SL/5/TrainingSamplesDCT_8_new.mat')

[rg, cg] = size(TrainsampleDCT_BG);
[rc, cc] = size(TrainsampleDCT_FG);

% Reading Image
img = imread('cheetah.bmp');
img = im2double(img);
[rows,cols] = size(img);
img_mask = imread('cheetah_mask.bmp');
img_mask = img_mask / 255;

% Using Matrix Representation
zigzags = zeros(rows,cols,64);
for row = 1:rows-7
    for col = 1:cols-7
        DCT = (dct2(img(row:row+7,col:col+7)));
        zigzag_matrix = zigzag(DCT);
        zigzag_matrix = zigzag_matrix';
        zigzags(row,col,:) = zigzag_matrix;
    end
end

BG_m = containers.Map();
BG_s = containers.Map();
BG_p = containers.Map();
FG_m = containers.Map();
FG_s = containers.Map();
FG_p = containers.Map();
C = 8;

% EM Training Phase
tic;
for i = 1:5
    [mu_grass, sigma_grass, pi_grass] = EM_learn(TrainsampleDCT_BG, C);
    [mu_cheetah, sigma_cheetah, pi_cheetah] = EM_learn(TrainsampleDCT_FG, C);
    BG_m(num2str(i)) = mu_grass;
    BG_s(num2str(i)) = sigma_grass;
    BG_p(num2str(i)) = pi_grass;
    FG_m(num2str(i)) = mu_cheetah;
    FG_s(num2str(i)) = sigma_cheetah;
    FG_p(num2str(i)) = pi_cheetah;
end
```

```
toc

% Compute Error
dim = [1, 2, 4, 8, 16, 24, 32, 40, 48, 56, 64];
errors = zeros(5, 5, 11);
for i = 1:5
    for j = 1:5
        res = zeros(rows, cols);
        for di = 1:size(dim, 2)
            d = dim(di);
            mu_BG = BG_m(num2str(i));
            mu_BG = mu_BG(:, 1:d);
            mu_FG = FG_m(num2str(j));
            mu_FG = mu_FG(:, 1:d);
            sigma_BG = BG_s(num2str(i));
            sigma_BG = sigma_BG(1:d, 1:d, :);
            sigma_FG = FG_s(num2str(j));
            sigma_FG = sigma_FG(1:d, 1:d, :);
            pi_BG = BG_p(num2str(i));
            pi_FG = FG_p(num2str(j));

            % Prior
            prior_BG = rg / (rc + rg);
            prior_FG = rc / (rc + rg);

            inv_BG = zeros(d, d, C);
            inv_FG = zeros(d, d, C);
            alpha_BG = zeros(1, C);
            alpha_FG = zeros(1, C);

            % Calculate Inverse Ahead
            for c = 1:C
                inv_BG(:, :, c) = pinv(sigma_BG(:, :, c));
                inv_FG(:, :, c) = pinv(sigma_FG(:, :, c));
                alpha_BG(1, c) = sum(log(diag(sigma_BG(:, :, c))));
                alpha_FG(1, c) = sum(log(diag(sigma_FG(:, :, c))));
            end

            % Calculate Error
            for row = 1:rows-7
                for col = 1:cols-7
                    zigzag_matrix = zigzags(row, col, :);
                    zigzag_matrix = zigzag_matrix(:);
                    zigzag_matrix = zigzag_matrix(1:d);
                    pBG = 0;
                    pFG = 0;
                    for c = 1:C
                        cur_mu_FG = mu_FG(c, :)';
                        cur_mu_BG = mu_BG(c, :)';
                        X_FG = zigzag_matrix - cur_mu_FG;
                        X_BG = zigzag_matrix - cur_mu_BG;
                        pBG = pBG + exp(log(pi_BG(c)) - 0.5 * X_BG' * inv_BG(:, :, c) *
                            X_BG - 0.5 * alpha_BG(c));
                        pFG = pFG + exp(log(pi_FG(c)) - 0.5 * X_FG' * inv_FG(:, :, c) *
```

```
                        X_FG - 0.5 * alpha_FG(c));
                    end
                    pFG = log(pFG) + log(prior_FG);
                    pBG = log(pBG) + log(prior_BG);
                    if pFG >= pBG
                        res(row, col) = 1;
                    end
                end
            end
            err = sum(sum(res ~= img_mask)) / (rows*cols)
            errors(i, j, di) = err;
        end
    end
end
```

```
% EM_learn.m
function [mu, sigma, pi] = EM_learn(dataset, C)
    [N, d] = size(dataset);

    % Randomized Initialization
    mu0 = zeros(C, d);
    sigma0 = zeros(d, d, C);
    index = randi(N, 1, C);
    for i = 1:C
        sigma0(:, :, i) = diag(ones(1, d) .* rand(1, d) + 1e-5);
        mu0(i, :) = dataset(index(i), :);
    end

    % EM Procedure
    pi0 = ones(1, C) / C;
    mu = mu0;
    sigma = sigma0;
    pi = pi0;
    pm = mu0(:, 1);
    for i = 1:5000
        % E-step solve H matrix
        H = zeros(N, C);
        for c = 1:C
            mu_c = mu0(c, :);
            sigma_c = sigma0(:, :, c);
            H(:, c) = mvnpdf(dataset, mu_c, sigma_c);
            H(:, c) = H(:, c) * pi0(c);
        end
        H = H ./ sum(H, 2);

        % M-step getting new parameters
        pi = sum(H) / N;
        for c = 1:C
            mu(c, :) = sum(H(:, c) .* dataset) / (sum(H(:, c)));
            x_c = dataset - mu0(c, :);
            s = sum((x_c .* x_c) .* H(:, c) / (sum(H(:, c)))) + 1e-5;
            sigma(:, :, c) = diag(s);
```

```
        end

        % Early Stopping
        if sum(abs(mu(:, 1) - pm ./ pm)) < 1e-2
            break;
        end

        mu0 = mu;
        sigma0 = sigma;
        pi0 = pi;
    end
end
```

```
% EM2.m
load('/Users/quan/Documents/MATLAB/SL/5/TrainingSamplesDCT_8_new.mat')

[rg, cg] = size(TrainsampleDCT_BG);
[rc, cc] = size(TrainsampleDCT_FG);

% Reading Image
img = imread('cheetah.bmp');
img = im2double(img);
[rows,cols] = size(img);
img_mask = imread('cheetah_mask.bmp');
img_mask = img_mask / 255;

% Using Matrix Representation
zigzags = zeros(rows,cols,64);
for row = 1:rows-7
    for col = 1:cols-7
        DCT = (dct2(img(row:row+7,col:col+7)));
        zigzag_matrix = zigzag(DCT);
        zigzag_matrix = zigzag_matrix';
        zigzags(row,col,:) = zigzag_matrix;
    end
end

BG_m = containers.Map();
BG_s = containers.Map();
BG_p = containers.Map();
FG_m = containers.Map();
FG_s = containers.Map();
FG_p = containers.Map();
Cs = [1, 2, 4, 8, 16, 32];

% EM Training Phase
tic;
for C = Cs
    [mu_grass, sigma_grass, pi_grass] = EM_learn(TrainsampleDCT_BG, C);
    [mu_cheetah, sigma_cheetah, pi_cheetah] = EM_learn(TrainsampleDCT_FG, C);
    BG_m(num2str(C)) = mu_grass;
    BG_s(num2str(C)) = sigma_grass;
```

```
    BG_p(num2str(C)) = pi_grass;
    FG_m(num2str(C)) = mu_cheetah;
    FG_s(num2str(C)) = sigma_cheetah;
    FG_p(num2str(C)) = pi_cheetah;
end
toc

% Compute Error
dim = [1, 2, 4, 8, 16, 24, 32, 40, 48, 56, 64];
errors = zeros(6, 11);
for C = Cs
    for di = 1:size(dim, 2)
        res = zeros(rows, cols);
        d = dim(di);
        mu_BG = BG_m(num2str(C));
        mu_BG = mu_BG(:, 1:d);
        mu_FG = FG_m(num2str(C));
        mu_FG = mu_FG(:, 1:d);
        sigma_BG = BG_s(num2str(C));
        sigma_BG = sigma_BG(1:d, 1:d, :);
        sigma_FG = FG_s(num2str(C));
        sigma_FG = sigma_FG(1:d, 1:d, :);
        pi_BG = BG_p(num2str(C));
        pi_FG = FG_p(num2str(C));

        % Prior
        prior_BG = rg / (rc + rg);
        prior_FG = rc / (rc + rg);

        inv_BG = zeros(d, d, C);
        inv_FG = zeros(d, d, C);
        alpha_BG = zeros(1, C);
        alpha_FG = zeros(1, C);

        % Calculate Inverse Ahead
        for c = 1:C
            inv_BG(:, :, c) = inv(sigma_BG(:, :, c));
            inv_FG(:, :, c) = inv(sigma_FG(:, :, c));
            alpha_BG(1, c) = sum(log(diag(sigma_BG(:, :, c))));
            alpha_FG(1, c) = sum(log(diag(sigma_FG(:, :, c))));
        end

        % Calculate Error
        for row = 1:rows-7
            for col = 1:cols-7
                zigzag_matrix = zigzags(row, col, :);
                zigzag_matrix = zigzag_matrix(:);
                zigzag_matrix = zigzag_matrix(1:d);
                pBG = 0;
                pFG = 0;
                for c = 1:C
                    cur_mu_FG = mu_FG(c, :)';
                    cur_mu_BG = mu_BG(c, :)';
                    X_FG = zigzag_matrix - cur_mu_FG;
```

```
                    X_BG = zigzag_matrix - cur_mu_BG;
                    pBG = pBG + exp(log(pi_BG(c)) - 0.5 * X_BG' * inv_BG(:, :, c) * X_BG - 0.5 * alpha_BG(c));
                    pFG = pFG + exp(log(pi_FG(c)) - 0.5 * X_FG' * inv_FG(:, :, c) * X_FG - 0.5 * alpha_FG(c));
                end
                pFG = log(pFG) + log(prior_FG);
                pBG = log(pBG) + log(prior_BG);
                if pFG >= pBG
                    res(row, col) = 1;
                end
            end
        end
        err = sum(sum(res ~= img_mask)) / (rows*cols)
        errors(C, di) = err;
    end
end
```