ECE 271A Quiz 5

Chen, Boyu / A59005520

Dec 3, 2021

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.

$$P_{X|Y}(x|i) = \sum_{c=1}^{C} \pi_c G(\mathbf{x}, \mu_c, \sigma_c^2)$$

where all σ_c^2 are diagonal matrices. We then apply the BDR based on these density estimates to the cheetah image and measure the probability of error as a function of the number of dimensions of the space (as before, use 1, 2, 4, 8, 16, 24, 32, 40, 48, 56, 64 dimensions).

To implement an EM algorithm, we first need to write down the likelihood of the complete data:

$$P_{X,Z}(x, z; \Psi) = P_{X|Z}(x|z; \Psi)P_Z(z; \Psi)$$

Considering the complete iid dataset $D_C = \{(x_1, z_1), ..., (x_N, z_N)\}$, with one-hot encoding and log-processing we can rewrite it as:

$$log P_{X,Z}(D, \{z_1, ..., z_N\}; \Psi) = \sum_{i,j} z_{ij} log[P_{X|Z}(x_i|e_j; \Psi)\pi_j]$$

Then for E-step, we could derive the Q function given observed data:

$$h_{ij} = P_{Z|X}(e_j|x_i; \Psi^{(n)})$$

$$Q(\Psi; \Psi^{(n)}) = \sum_{i,j} h_{ij} log[P_{X|Z}(x_i|e_j; \Psi)\pi_j]$$

In M-step, we can solve the maximization, deriving a cloesd-form solution if there is one:

$$\Psi^{n+1} = arg \max_{\Psi} \sum_{i,j} h_{ij} log[P_{X|Z}(x_i|e_j; \Psi)\pi_j]$$

Especially, for Gaussian mixtures, in the E-step we could do:

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

And in M-step, we could solve the maximization as:

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

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

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

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). The plots of probability of error vs. dimension for each of the 25 classifiers are shown as below.

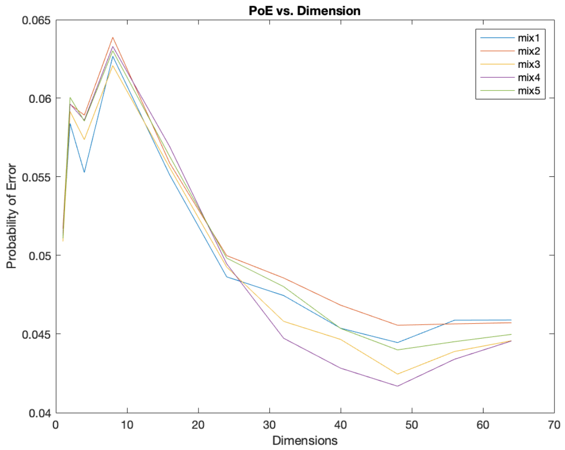

Figure 1: PoE for mixture 1-5

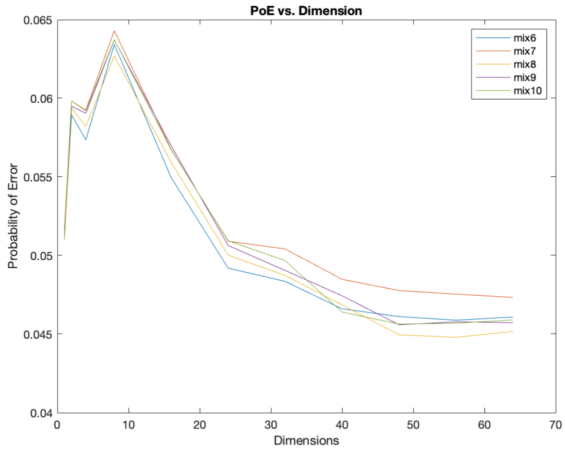

Figure 2: PoE for mixture 6-10

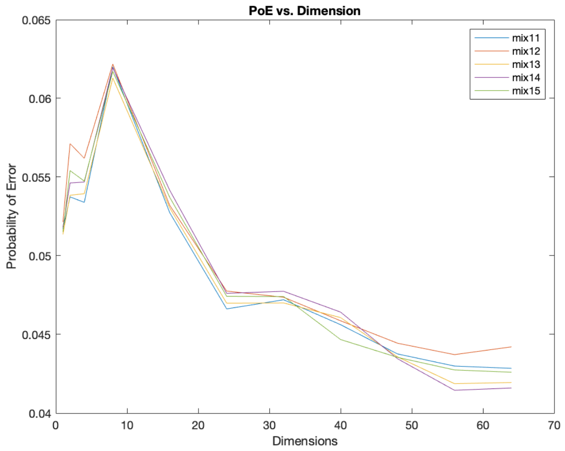

Figure 3: PoE for mixture 11-15

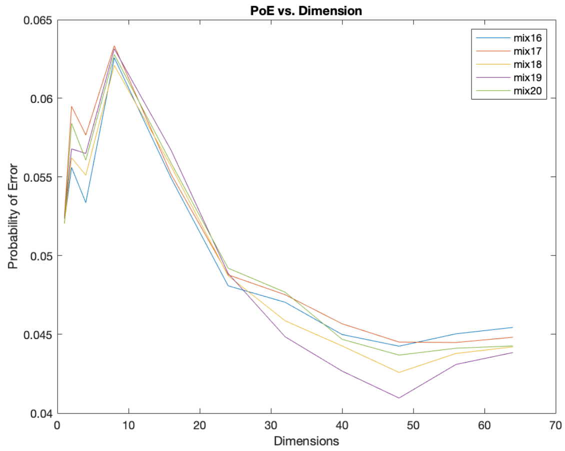

Figure 4: PoE for mixture 16-20

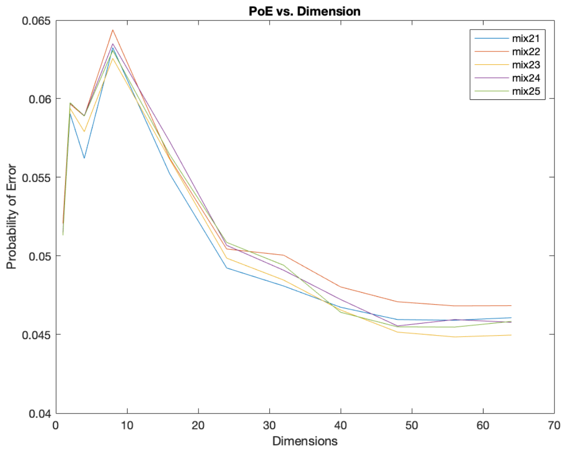

Figure 5: PoE for mixture 21-25

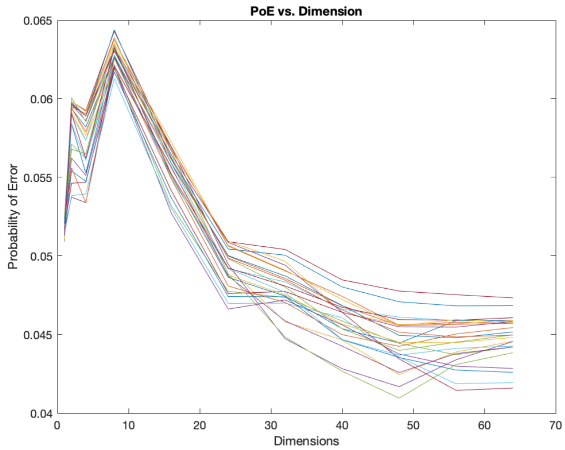

Figure 6: PoE for all mixtures

Analysis:

I think that PoE is highly dependent on the initialization.

At the first glace we observe that all params mixtures perform differently on all feature dimensions. This is because they are assigned with different initialization during E-step. EM is all about iteration and optimization. In Gaussian mixture cases, EM algorithm should maximize the likelihood function with parameters θ with respect to θ, which are μ_c, σ_c^2 and π_c for each mixture component. In general, assuming that the function might go up and down and have multiple local maximums which we cannot know exactly, by iteration the result might not be a global optima but a **local** one. As the models are assigned with different random initial value, they will then likely to reach different local optimal areas, resulting in different parameters.

Also, we notice that the trends for different mixture pairs are very similar. I suppose this is because the information provided by same dimensions are the same, resulting in similar prediction error.

b) For each class, learn mixtures with C ∈ {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?

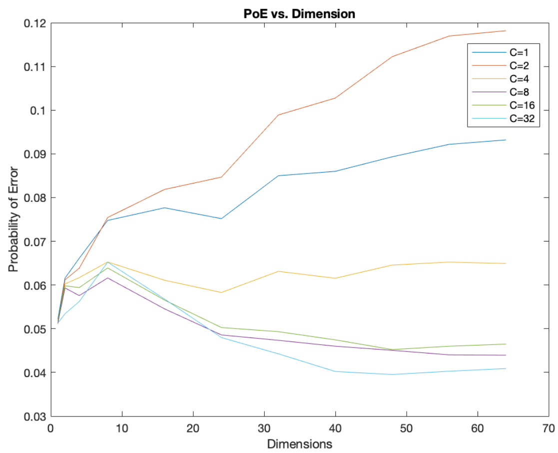

Figure 7: PoE for C ∈ {1, 2, 4, 8, 16, 32}

Analysis:

Through observation, we notice that the PoE for $C = 1, 2, 4$ increases as dimension goes up. In cases where $C = 8, 16, 32$, PoE decreases when dimension goes larger. Under multi-Gaussian model assumption, one explanation could be that with small numbers of components the model cannot describe the distribution well enough. By increasing the number of components, we can take more models into consideration to make prediction, which contains more information and would result in lower PoE and better performance.

However, this is not just to claim that more mixture components are always better. For one thing, as the number of components grows, the program would cost a lot more time to execute. For another thing, more mixed models might result in overfitting, which would then reduce the accuracy.

Matlab Code:

- main.m

```matlab
%% load data
load("TrainingSamplesDCT_8_new.mat");

pattern = dlmread("Zig-Zag Pattern.txt") + 1;

[cheetah, cheetacolorhmap] = imread("cheetah.bmp");

cheetah2 = im2double(cheetah);

cheetah2_temp = cheetah2;

cheetah2_temp(262,277) = 0;

[cheetah_mask, cheetah_mask_colormap] = imread("cheetah_mask.bmp");
cheetah_mask = im2double(cheetah_mask/255);

[rowGrass, ¬] = size(TrainsampleDCT_BG);
[rowCheetah, ¬] = size(TrainsampleDCT_FG);
p_bg = rowGrass / (rowCheetah + rowGrass);
p_fg = rowCheetah / (rowCheetah + rowGrass);

feature_dim = 64;

dct_vec = zeros(255*270, 64);
for i = 1 : 255
    for j = 1 : 270
        block = cheetah2_temp(i:i+7, j:j+7);
        block2vec = dct2(block);
        dct_vec(((i-1)*270+j),:) = vec2zigzag(pattern, block2vec);
    end
end

dims_lst = [1 2 4 8 16 24 32 40 48 56 64];
[¬,dims_num] = size(dims_lst);
n_class = 8;
n_mix = 5;

mu_fg = zeros(n_mix, feature_dim*n_class);
sigma_fg = zeros(n_mix, feature_dim*n_class);
pi_fg = zeros(n_mix, n_class);
mu_bg = zeros(n_mix, feature_dim*n_class);
sigma_bg = zeros(n_mix, feature_dim*n_class);
pi_bg = zeros(n_mix, n_class);

%% training
fprintf('Loading completed... Starting Training.\n')

for mix = 1:n_mix
    [mu_bg(mix,:), sigma_bg(mix,:), pi_bg(mix,:)] = em(TrainsampleDCT_BG, n_class, ...
        feature_dim, 200);
    [mu_fg(mix,:), sigma_fg(mix,:), pi_fg(mix,:)] = em(TrainsampleDCT_FG, n_class, ...
        feature_dim, 200);
end

save('em_1_data.mat');
%% prediction and poe calculating

poe = zeros(n_mix*n_mix, dims_num);

for mix1 = 1:n_mix
    for mix2 = 1:n_mix
        for dim = 1:dims_num
            mix_mask = BDR(dct_vec,dims_lst(dim),...
                mu_fg(mix1,:),mu_bg(mix2,:),sigma_fg(mix1,:),sigma_bg(mix2,:),...
                pi_fg(mix1,:),pi_bg(mix2,:),p_fg,p_bg,255,270,n_class);
            poe((mix1-1)*5+mix2,dim) = Error(mix_mask,p_fg,p_bg);
        end
    end
end

```

```
%% save data
save('em1_1_finaldata.mat')

%% subplots
for i = 1:5
    figure;
    plot(dims_lst,poe((i-1)*5+1:(i-1)*5+5,:)');
    hold on
    for j = 1:5
        leg_str{j} = ['mix', num2str(5*(i-1)+j)];
    end
    legend(leg_str)
    title('PoE vs. Dimension')
    xlabel('Dimensions')
    ylabel('Probability of Error')
end
%% plot all mixture pair in one pic
plot(dims_lst,poe')
for j = 1:25
    leg_str{j} = ['mix', num2str(j)];
end
% legend(leg_str)
title('PoE vs. Dimension')
xlabel('Dimensions')
ylabel('Probability of Error')
```

- main2.m

```
%% Load data and define parameters.
load('em1_1_finaldata.mat'); % MATLAB data array that contains information already ... loaded in the first section of em_main.

dims_lst = [1 2 4 8 16 24 32 40 48 56 64]; % Desired dimensions.
[¬,dims] = size(dims_lst);
classes = [1 2 4 8 16 32];
n_classes = size(classes,2);
poe2 = zeros(n_classes,dims);

%% EM model training and prediction.
for class = 1:n_classes
   mix_mask2 = zeros(255,270);
   [mu_fg_c,sigma_fg_c,pi_fg_c] = em(TrainsampleDCT_FG,classes(class),M,200);
   [mu_bg_c,sigma_bg_c,pi_bg_c] = em(TrainsampleDCT_BG,classes(class),M,200);
   for dim = 1:dims
       mix_mask2 = BDR(dct_vec,dims_lst(dim),mu_fg_c,mu_bg_c,sigma_fg_c,sigma_bg_c,...
           pi_fg_c,pi_bg_c,p_fg,p_bg,255,270,classes(class));
       poe2(class,dim) = Error(mix_mask2,p_fg,p_bg);
   end
end

%% plot poe vs dimensions

plot(dims_lst,poe2');
legend('C=1','C=2','C=4','C=8','C=16','C=32') %1 2 4 8 16 32
title('PoE vs. Dimension')
xlabel('Dimensions')
ylabel('Probability of Error')
```

- em.m

```
function [mu, sigma, pi_c] = em(dct_vec, n_class, dct_dim, num_iter)
    [n_rows, ¬] = size(dct_vec);
    dct_vec_dim = dct_vec(:,1:dct_dim);

    sigma_c = diag(diag(2*rand(dct_dim*n_class,dct_dim*n_class)+2));
    mu_c = 3*rand(n_class, dct_dim)+3;
    pi_c = (randi(20, 1, n_class));
    pi_c = pi_c/sum(pi_c);
    epsilon = (1e-06)*ones(size(sigma_c));
    z = zeros(n_rows, n_class);

```

```matlab
    for iter = 1:num_iter
        % E-step
        z_pre = z;
        for row = 1:n_rows
            p_x = zeros(1, n_class);
            for comp = 1:n_class
                slide1 = (comp-1)*dct_dim;
                slide2 = comp*dct_dim;
                sigma = sigma_c(slide1+1:slide2, slide1+1:slide2);
                mu = mu_c(comp,:);
                p_x(comp) = mvnpdf(dct_vec_dim(row,:),mu,sigma)*pi_c(comp);
            end
            z(row,:) = p_x/sum(p_x);
        end
        pi_c = sum(z,1)/n_rows;

        % M-step
        for comp = 1:n_class
            slide1 = (comp-1)*dct_dim;
            slide2 = comp*dct_dim;
            sig = (dct_vec_dim-repmat(mu_c(comp,:),n_rows,1));
            sigma = sig.*(repmat(z(:,comp),1,dct_dim));
            tot = sum(z(:,comp));
            sigma_c(slide1+1:slide2, slide1+1:slide2) = (sigma'*sig)/tot;
            mu_c(comp,:) = sum(dct_vec_dim.*repmat(z(:,comp),1,dct_dim))/tot;
        end
        sigma_c = diag(diag(sigma_c + epsilon));
        if (log(z) - log(z_pre)) < 1e-06
            break;
        end
    end

    % return the final params
    mu = zeros(1, dct_dim*n_class);
    for comp = 1:n_class
        mu((comp-1)*dct_dim+1:comp*dct_dim) = mu_c(comp,:);
    end
    sigma = diag(sigma_c).';
end
```

- BDR.m

```matlab
function [mask] = BDR(dct_vec,dim,mu_fg,mu_bg,sigma_fg,sigma_bg,...
    pi_fg,pi_bg,p_fg,p_bg,rows,cols,n_class)
    mask = zeros(rows,cols);
    k=1;
    for x = 1:rows
        for y = 1:cols
            vec = dct_vec(k,:);
            k=k+1;
            if (p_fg*Prob(vec, dim, n_class, mu_fg, sigma_fg, pi_fg) > ...
                    p_bg*Prob(vec, dim, n_class, mu_bg, sigma_bg, pi_bg))
                mask(x,y) = 1;
            end
        end
    end
end
```

- Prob.m

```matlab
function result = Prob(vec, dim, n_class, mu, sigma, pi)
    result = 0;
    vec = vec(:,1:dim);
    for c=1:n_class
        slideStart = 64*(c-1);
        mu_temp = mu(slideStart+1:slideStart+dim);
        sigma_temp = diag(sigma(slideStart+1:slideStart+dim));
        result = result + pi(c) * mvnpdf(vec, mu_temp, sigma_temp);
    end
end
```

- Error.m

```
function [poe] = Error(A,p_fg,p_bg)
    cheetah_mask = imread('cheetah_mask.bmp');
    cheetah_mask = cheetah_mask == 255;

    cheetah_mask_one = find(cheetah_mask);
    cheetah_mask_zero = find(¬cheetah_mask);

    num_incorrect_zero = sum(A(cheetah_mask_zero) == 1);
    num_incorrect_one = sum(A(cheetah_mask_one) == 0);

    poe = (num_incorrect_zero*p_bg)/sum(sum(cheetah_mask == 0)) +...
        (num_incorrect_one*p_fg)/sum(sum(cheetah_mask == 1));
end
```

- vec2zigzag.m

```
function zgvec = vec2zigzag(pattern,vec)
    zgvec = zeros(1,64);
    for i=1:8
        for j=1:8
         zgvec(pattern(i,j)) = vec(i,j);
        end
    end
end
```