

## ECE 271A Homework 5

1. 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 x 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.  $P_{X|Y}(x|i) = \sum_{c=1}^C \pi_c G(x, \mu_c, \sigma_c)$  where all  $\sigma_c$  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  $\{1, 2, 4, 8, 16, 24, 32, \dots, 64\}$  dimensions).
  - (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.

### *Solution.*

In order to perform EM for Gaussian mixtures, we need three variables, that are:  $\pi$ ,  $\mu$ , and  $\Sigma$ . The first step is to initialize all the variables. Firstly, we create the parameter  $\pi$ . Creating it is easy, we just create a matrix of size  $C$ , each entry is less than 1, and the sum of the matrix equals to 1. Secondly, we create the parameter  $\mu$ . We randomly select  $C$  observations from the training data set. Lastly, we create a positive definite matrix for the parameter  $\Sigma$ . For simplicity, we create a random vector and diagonalize it and make it our  $\Sigma$ . After setting up the parameters, we now move on to the E step of our learning. We calculate the  $h_{ij}$  variable by using the formula mentioned below.

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

With the  $h_{ij}$  value known, we can proceed to the M step.  $\psi^{(n+)}$  can be calculated by using the formula mentioned below.

$$\psi^{(n+)} = \arg \min_{\psi} \sum_{ij} h_{ij} \log[G(x_i, u_j^{(n)}, \sigma_j^{(n)})] \quad (2)$$

We want to optimize the formula by using Lagrangian (per the note in the lecture). The formula we need for updating our parameters are listed below.

$$u_j^{(n+1)} = \frac{\sum_i h_{ij} x_i}{\sum_i h_{ij}} \quad (3)$$

$$\pi_j^{(n+1)} = \frac{1}{n} \sum_i h_{ij} \quad (4)$$

$$\sigma_j^{2(n+1)} = \frac{\sum_i h_{ij} (x_i - u_j)^2}{\sum_i h_{ij}} \quad (5)$$

With those formulae in hand, we have no problem updating our parameters. We set a maximum number of time for update, **EMLimit** = 1000. If the number of iteration exceeds it, then our algorithm will break. Also, we compare our result from the current iteration with that of the last iteration. If the value of the likelihood does not change by 0.001, then we will break our algorithm as well. Since there is no sense running the algorithm if the value does not differ. One crucial thing to keep in mind is that  $\Sigma$  should always be positive definite; otherwise, the algorithm will crash. To achieve this, we add a small value to  $\Sigma$  after each iteration. By doing this, we can assure that  $\Sigma$  remains positive definite at all time.

Now **Part a** asks us to train 5 EMs for *cheetah* and 5 EMs for *grass* and then plot the error rates and compare them, i.e., perform a cross-comparison among those 25 EMs. It is an arduous task but after hours of running the algorithm, here are my results. I plot the first EM of *cheetah* with five EMs of *grass*, then plot the second EM of *cheetah* with five EMs of *grass*, etc. From **Figure 1 (a)** to **Figure 1 (k)**, we can confidently say that as the value of dimensions increase, so does the performance. We can be more precise by observing the plot from **Figure 2** to **Figure 6**. The error rates decrease dramatically. There exhibit similar results across all those 25 EMs (5 EMs for *cheetah* and 5 for *grass*).

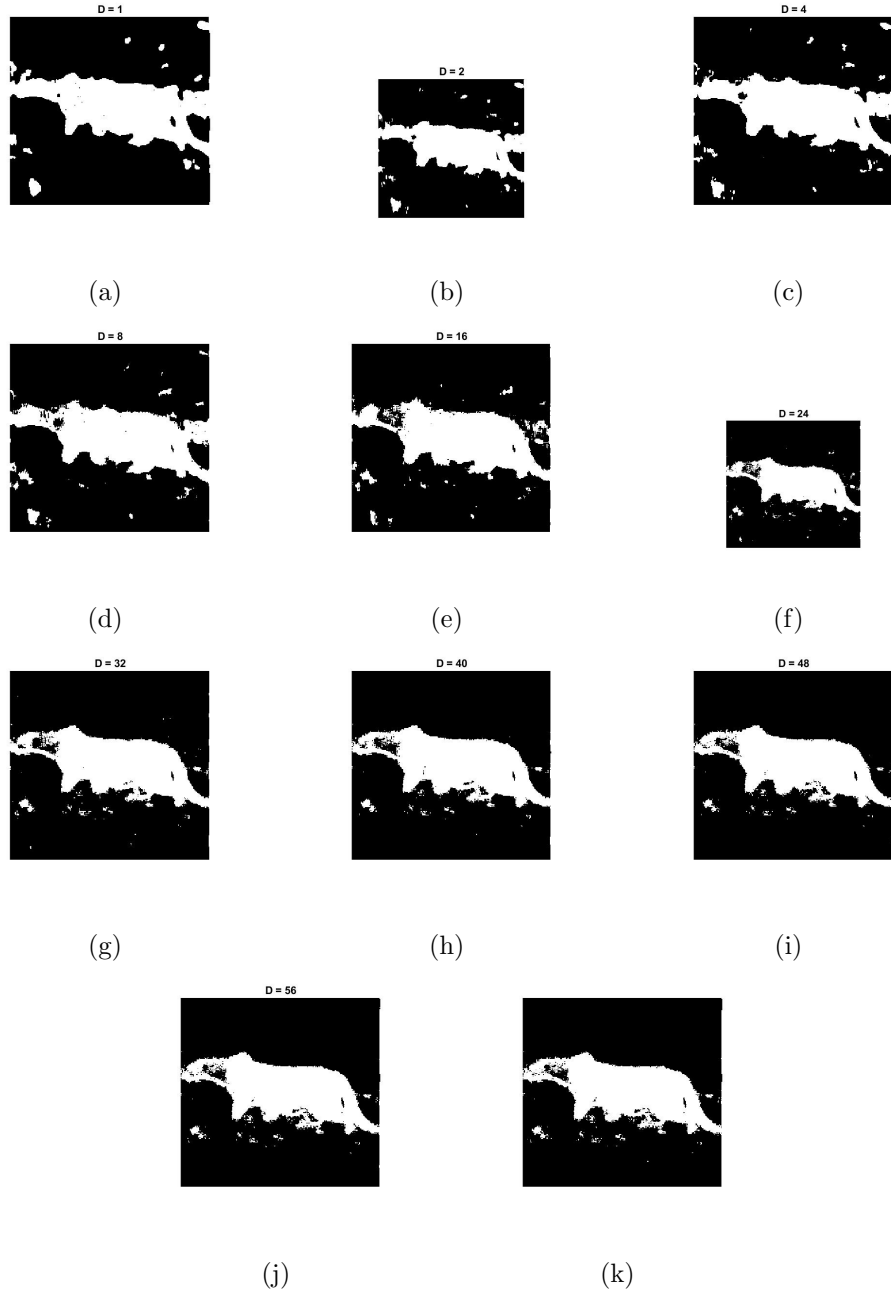


Figure 1: Results from EM: (a)  $D = 1$  (b)  $D = 2$  (c)  $D = 4$  (d)  $D = 8$  (e)  $D = 16$  (f)  $D = 24$  (g)  $D = 32$  (h)  $D = 40$  (i)  $D = 48$  (j)  $D = 56$  (k)  $D = 64$

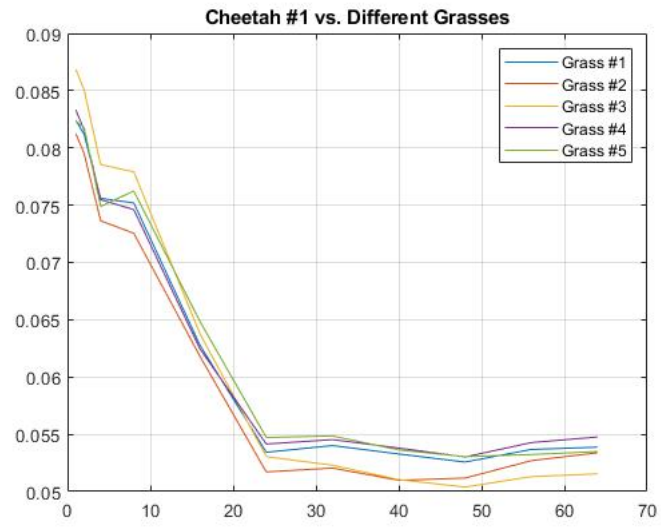


Figure 2: Cheetah #1 vs. Grasses

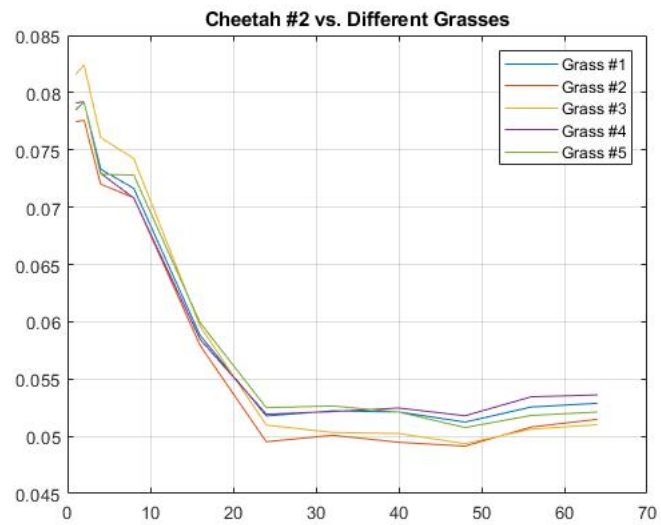


Figure 3: Cheetah #2 vs. Grasses

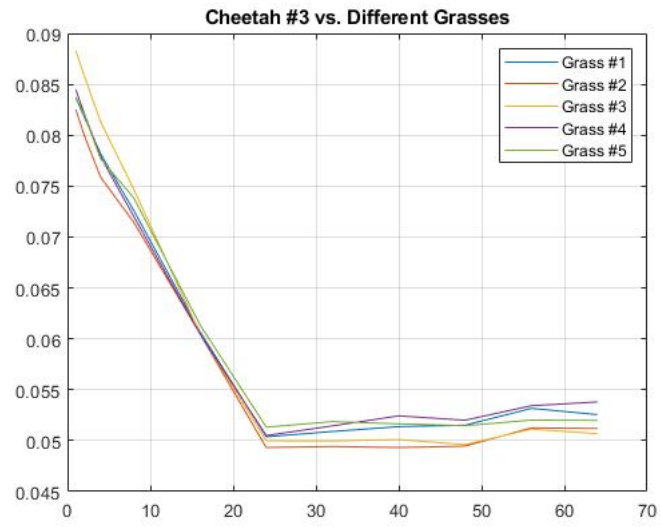


Figure 4: Cheetah #3 vs. Grasses

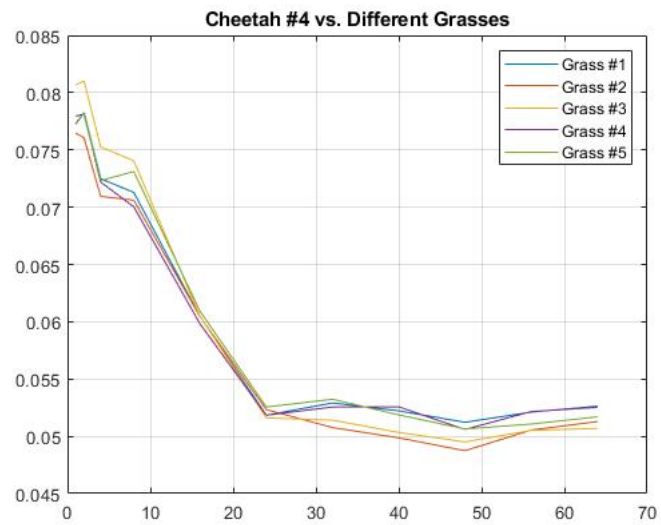


Figure 5: Cheetah #4 vs. Grasses

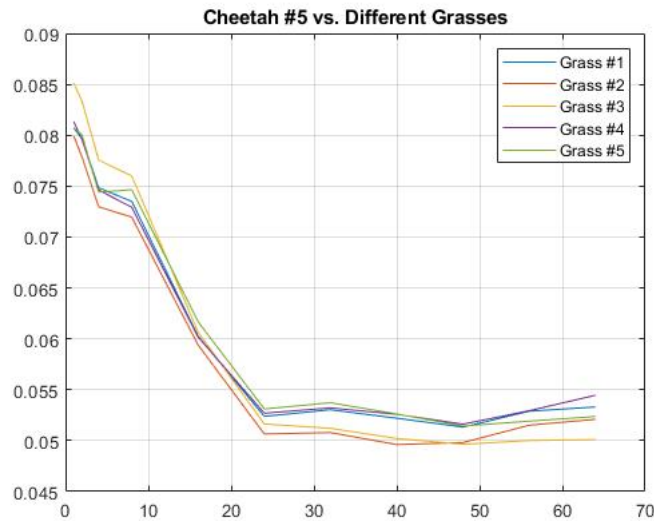


Figure 6: Cheetah #5 vs. Grasses

- (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?

*Solution.*

After learning mixture models for  $C = 1, 2, 4, 8, 16, 32$ , we plot the result as shown in **Figure 7**. We can tell that when  $C = 1$  and  $C = 2$ , the error rates are high. Nevertheless, when we look at  $C$  that are greater than or equal to 4, the EM algorithm yields decent results. And the results from all of them are quiet similar when the value of the dimensions is low. They only perform slightly different when the value of the dimensions is high, i.e., when  $d$  is greater than or equal to 32.

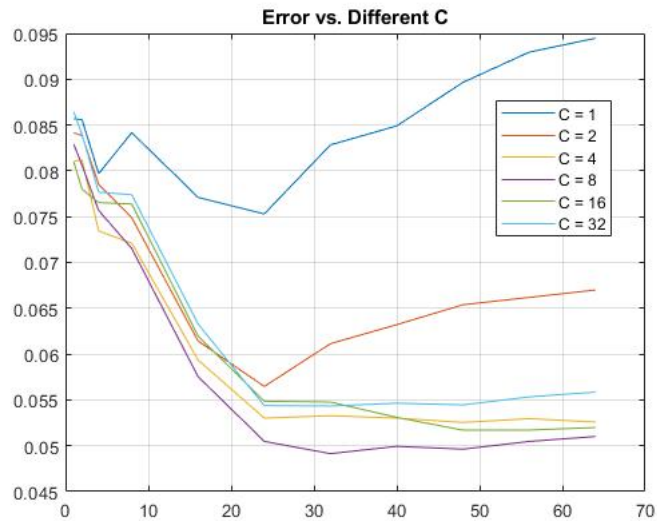


Figure 7: Error Rate vs. Different Dimensions

## Code for EM (Mother Code)

```

1 clear
2 clc
3
4 % load all the data
5 load("ZigZagVec.mat") % 1 x 64 vector
6 load("cheetahMat.txt") % our image
7 load("TrainingSamplesDCT_8_new.mat") % the training data
  sets
8 load("zigzagImg.mat") % the preprocessed image
9 load("cheetah_mask.mat") % the preprocessed mask
10
11 % dimensions for the training & the parameters for EM
12 C = 8; % number of components
13 dimensionList = [1,2,4,8,16,24,32,40,48,56,64]; % number of
  dimensions of the space
14
15 % # of samples
16 [n_cheetah, ~] = size(TrainsampleDCT_FG);
17 [n_grass, ~] = size(TrainsampleDCT_BG);
18 totalSamples = (n_cheetah + n_grass);
19
20 % get the size of the image

```

```

21 [m, n] = size(cheetahMat);
22
23 % create a matrix
24 m = m - 7;
25 n = n - 7;
26 totalPixels = m * n;
27
28 % —— EM for cheetah ——
29
30 % initialize pi by creating a random normalized 1 x C matrix
31 pi_cheetah = randi(1, C);
32 pi_grass    = randi(1, C);
33 pi_cheetah = pi_cheetah / sum(pi_cheetah);
34 pi_grass    = pi_grass    / sum(pi_grass);
35
36 % initialize mu by choosing C samples randomly from the
    training data set
37 p = randperm(n_cheetah, C); % pick C random numbers from 0
    to the size of the sample of cheetah
38 mu_cheetah = zeros(C, 64); % create an array for mu
39 mu_grass    = zeros(C, 64); % create an array for mu
40 for i = 1 : C
41     mu_cheetah(i, :) = TrainsampleDCT_FG(p(i), 64);
42     mu_grass(i, :)   = TrainsampleDCT_BG(p(i), 64);
43 end
44
45 % initialize sigma by creating a diagonal matrix with random
    variable
46 sigma_cheetah = zeros(64, 64, C);
47 sigma_grass    = zeros(64, 64, C);
48 for i = 1 : C
49     randomVec = rand(1, 64); % generate a random vector (1
        x 64)
50     sigma_cheetah(:, :, i) = diag(randomVec); % diagonalize
        the random vector
51     randomVec = rand(1, 64); % generate a random vector (1
        x 64)
52     sigma_grass(:, :, i) = diag(randomVec); % diagonalize the
        random vector

```



```

53 end
54
55 EMLimit = 1000; % maximum iterations
56
57 % —— Start of EM for cheetah ——
58 jointPro_cheetah = zeros(n_cheetah, C);
59 for i = 1 : EMLimit
60     % E step
61     for j = 1 : C
62         %  $P_{\{X, Z\}}(x, z; \psi) = P_{\{X|Z\}}(x|z; \psi) * P_Z(z; \psi)$ 
63         jointPro_cheetah(:, j) = mvnpdf(TrainsampleDCT_FG,
            mu_cheetah(j, :), sigma_cheetah(:, :, j)) *
            pi_cheetah(j); % mvnpdf returns the pdf of given X,
            mu, sigma
64     end
65
66     % calculate the log likelihood of the
67     sumRow = sum(jointPro_cheetah, 2); % returns the sum of
        the elements in each row
68     likelihood_cheetah = sum(log(sumRow));
69
70     %  $h_{ij} = (G(x_i, u_j, \sigma_j) * \pi_j) / \sum(G(x_i, u_j, \sigma_j) * \pi_j)$ 
71     hij_cheetah = jointPro_cheetah ./ sumRow;
72
73     % M step
74
75     % update pi for (n+1)
76     %  $\pi = 1 / n * \sum(h_{ij})$ 
77     pi_cheetah = sum(hij_cheetah) / n_cheetah;
78
79     % update mu for (n+1)
80     %  $\mu = \sum(h_{ij} * x) / \sum(h_{ij})$ 
81     % TODO: mu_cheetah changes dimension
82     mu_cheetah = (hij_cheetah' * TrainsampleDCT_FG) ./ sum(
        hij_cheetah)';
83
84     % update sigma for (n+1)

```

```

85 % sigma^2 = sum(hij * (x - mu)^2) / sum(hij)
86 for j = 1:C
87     sigma_cheetah(:, :, j) = diag(diag(((TrainsampleDCT_FG -
88         mu_cheetah(j, :))' * hij_cheetah(:, j) * ...
89         (TrainsampleDCT_FG - mu_cheetah(j, :)) ./ sum(
90             hij_cheetah(:, j), 1)) + 0.0000001));
91 end
92 % break condition
93 % breaks the loop if the likelihood does not change more
94 % than 0.001
95 if i > 1
96     if abs(likelihood_cheetah -
97         likelihood_cheetah_previous) < 0.001
98         break;
99     end
100 end
101 % store the current result as the previous
102 likelihood_cheetah_previous = likelihood_cheetah;
103 end
104 % — End of EM for cheetah —
105 % — Start of EM for grass —
106 jointPro = zeros(n_grass, C);
107 for i = 1 : EMLimit
108     % E step
109     for j = 1 : C
110         %  $P_{\{X, Z\}}(x, z; \psi) = P_{\{X|Z\}}(x|z; \psi) * P_Z(z; \psi)$ 
111         jointPro(:, j) = mvnpdf(TrainsampleDCT_BG, mu_grass(j,
112             :), sigma_grass(:, :, j)) * pi_grass(j); % mvnpdf
113         % returns the pdf of given X, mu, sigma
114     end
115 end
116 % calculate the log likelihood of the
117 sumRow = sum(jointPro, 2); % returns the sum of the
118 % elements in each row

```

```

116     likelihood_grass = sum(log(sumRow));
117
118     % h_ij = (G(x_i, u_j, sigma_j) * pi_j) / sum(G(x_i, u_j,
119         sigma_j) * pi_j)
120     hij_grass = jointPro ./ sumRow;
121
122     % M step
123
124     % update pi for (n+1)
125     % pi = 1 / n * (sum(hij))
126     pi_grass = sum(hij_grass) / n_grass;
127
128     % update mu for (n+1)
129     % mu = sum(hij * x)/sum(hij)
130     mu_grass = hij_grass' * TrainsampleDCT_BG ./ sum(hij_grass
131         )';
132
133     % update sigma for (n+1)
134     % sigma^2 = sum(hij * (x - mu)^2) / sum(hij)
135     for j = 1:C
136         sigma_grass(:, :, j) = diag(diag(((TrainsampleDCT_BG -
137             mu_grass(j, :))' .* hij_grass(:, j))' * ...
138             (TrainsampleDCT_BG - mu_grass(j, :)) ./ sum(
139                 hij_grass(:, j), 1)) + 0.0000001));
140     end
141
142     % break condition
143     % breaks the loop if the likelihood does not change more
144     % than 0.001
145     if i > 1
146         if abs(likelihood_grass - likelihood_grass_previous) <
147             0.001
148             break;
149         end
150     end
151
152     % store the current result as the previous
153     likelihood_grass_previous = likelihood_grass;
154
155

```

```

149 end
150 % —— End of EM for grass ——
151
152 % —— BDR ——
153 lenList = length(dimensionList);
154 errorMat = zeros(1, lenList);
155 for curDim = 1 : lenList
156
157     Kth = dimensionList(curDim);
158
159     % compare BDR for EM
160     maskVec = zeros(m * n, 1);    % a vector of our mask, will
        resize it later
161
162     for x = 1 : length(zigzagImg)
163
164         % set the probability of each class as zero
165         pro_cheetah = 0;
166         pro_grass    = 0;
167
168         % compute total BDR for cheetah
169         for y = 1:size(mu_cheetah,1)
170             pro_cheetah = pro_cheetah + mvnpdf(zigzagImg(x, 1
                : Kth), mu_cheetah(y, 1 : Kth), sigma_cheetah(1
                : Kth, 1 : Kth,y)) * pi_cheetah(y);
171         end
172
173         % compute total BDR for grass
174         for y = 1:size(mu_grass,1)
175             pro_grass = pro_grass + mvnpdf(zigzagImg(x, 1 :
                Kth), mu_grass(y, 1 : Kth), sigma_grass(1 : Kth,
                1 : Kth,y))*pi_grass(y);
176         end
177
178         % decide whether the pixel is cheetah or grass
179         if pro_cheetah > pro_grass
180             maskVec(x) = 1;
181         end
182     end

```

```

183
184     % resize the vector to matrix
185     maskMat = Vec2Mat(maskVec);
186
187     % compute the error rate
188     errorMat(curDim) = Err(cheetah_mask, maskMat);
189     errorRate = errorMat ./ (m * n);
190     imshow(maskMat)
191     figure
192 end

```

## Code for Part A

```

1 clear
2 clc
3
4 % load all the data
5 load("ZigZagVec.mat") % 1 x 64 vector
6 load("cheetahMat.txt") % our image
7 load("TrainingSamplesDCT_8_new.mat") % the training data
   sets
8 load("zigzagImg.mat") % the preprocessed image
9 load("cheetah_mask.mat") % the preprocessed mask
10
11 % dimensions for the training & the parameters for EM
12 C = 8; % number of components
13 dimensionList = [1,2,4,8,16,24,32,40,48,56,64]; % number of
   dimensions of the space
14
15 % # of samples
16 [n_cheetah, ~] = size(TrainsampleDCT_FG);
17 [n_grass, ~] = size(TrainsampleDCT_BG);
18 totalSamples = (n_cheetah + n_grass);
19
20 % get the size of the image
21 [m, n] = size(cheetahMat);
22
23 % create a matrix
24 m = m - 7;

```

```

25 n = n - 7;
26 totalPixels = m * n;
27
28 % —— EM for cheetah ——
29
30 % —— End of EM for cheetah ——
31
32 % —— Start of EM for grass ——
33
34 % —— End of EM for grass ——
35
36 pi_cheetah = zeros(1, 8, 5);
37 pi_grass    = zeros(1, 8, 5);
38 mu_cheetah = zeros(8, 64, 5);
39 mu_grass    = zeros(8, 64, 5);
40 sigma_cheetah = zeros(64, 64, 8, 5);
41 sigma_grass = zeros(64, 64, 8, 5);
42
43 for i = 1 : 5
44     [pi_cheetah(:, :, i), mu_cheetah(:, :, i), sigma_cheetah
         (:, :, :, i)] = GenerateCheetah(n_cheetah,
         TrainsampleDCT_FG);
45     [pi_grass(:, :, i), mu_grass(:, :, i), sigma_grass(:, :,
         :, i)] = GenerateGrass(n_grass, TrainsampleDCT_BG);
46 end
47
48 % —— BDR ——
49 errorRate1 = zeros(1, 11, 5);
50 for i = 1 : 5
51     errorRate1(:, :, i) = ComputeBDR(zigzagImg, cheetah_mask,
         mu_cheetah(:, :, 1), sigma_cheetah(:, :, :, 1),
         pi_cheetah(:, :, 1), mu_grass(:, :, i), sigma_grass(:,
         :, :, i), pi_grass(:, :, i));
52 end
53 errorRate2 = zeros(1, 11, 5);
54 for i = 1 : 5
55     errorRate2(:, :, i) = ComputeBDR(zigzagImg, cheetah_mask,
         mu_cheetah(:, :, 2), sigma_cheetah(:, :, :, 2),
         pi_cheetah(:, :, 2), mu_grass(:, :, i), sigma_grass(:,

```

```

        :, :, i), pi_grass(:, :, i));
56 end
57 errorRate3 = zeros(1, 11, 5);
58 for i = 1 : 5
59     errorRate3(:, :, i) = ComputeBDR(zigzagImg, cheetah_mask,
        mu_cheetah(:, :, 3), sigma_cheetah(:, :, :, 3),
        pi_cheetah(:, :, 3), mu_grass(:, :, i), sigma_grass(:,
        :, :, i), pi_grass(:, :, i));
60 end
61 errorRate4 = zeros(1, 11, 5);
62 for i = 1 : 5
63     errorRate4(:, :, i) = ComputeBDR(zigzagImg, cheetah_mask,
        mu_cheetah(:, :, 4), sigma_cheetah(:, :, :, 4),
        pi_cheetah(:, :, 4), mu_grass(:, :, i), sigma_grass(:,
        :, :, i), pi_grass(:, :, i));
64 end
65 errorRate5 = zeros(1, 11, 5);
66 for i = 1 : 5
67     errorRate5(:, :, i) = ComputeBDR(zigzagImg, cheetah_mask,
        mu_cheetah(:, :, 5), sigma_cheetah(:, :, :, 5),
        pi_cheetah(:, :, 5), mu_grass(:, :, i), sigma_grass(:,
        :, :, i), pi_grass(:, :, i));
68 end
69
70
71 % —— functions ——
72
73 % generate a class for cheetah
74 function [pi_cheetah, mu_cheetah, sigma_cheetah] =
    GenerateCheetah(n_cheetah, TrainsampleDCT_FG)
75     C = 8;
76     pi_cheetah = randi(1, C);
77     pi_cheetah = pi_cheetah / sum(pi_cheetah);
78     p = randperm(n_cheetah, C);
79     mu_cheetah = zeros(C, 64);
80     for i = 1 : C
81         mu_cheetah(i, :) = TrainsampleDCT_FG(p(i), 64);
82     end
83     sigma_cheetah = zeros(64, 64, C);

```

```

84     for i = 1 : C
85         randomVec = rand(1, 64);
86         sigma_cheetah(:, :, i) = diag(randomVec);
87     end
88     EMLimit = 1000;
89     jointPro_cheetah = zeros(n_cheetah, C);
90     for i = 1 : EMLimit
91         for j = 1 : C
92             jointPro_cheetah(:, j) = mvnpdf(TrainsampleDCT_FG,
93                 mu_cheetah(j, :), sigma_cheetah(:, :, j)) *
94                 pi_cheetah(j); % mvnpdf returns the pdf of given
95                 X, mu, sigma
96         end
97         sumRow = sum(jointPro_cheetah, 2);
98         likelihood_cheetah = sum(log(sumRow));
99         hij_cheetah = jointPro_cheetah ./ sumRow;
100        pi_cheetah = sum(hij_cheetah) / n_cheetah;
101        mu_cheetah = (hij_cheetah' * TrainsampleDCT_FG) ./ sum(
102            hij_cheetah)';
103        for j = 1:C
104            sigma_cheetah(:, :, j) = diag(diag(((
105                TrainsampleDCT_FG - mu_cheetah(j, :))' .*
106                hij_cheetah(:, j))' * ...
107                (TrainsampleDCT_FG - mu_cheetah(j, :)) ./ sum(
108                    hij_cheetah(:, j), 1)) + 0.0000001));
109        end
110        if i > 1
111            if abs(likelihood_cheetah -
112                likelihood_cheetah_previous) < 0.001
113                break;
114            end
115        end
116        likelihood_cheetah_previous = likelihood_cheetah;
117    end
118 end
119
120 % generate a class for grass
121 function [pi_grass, mu_grass, sigma_grass] = GenerateGrass(
122     n_grass, TrainsampleDCT_BG)

```



```

114     C = 8;
115     pi_grass = randi(1, C);
116     pi_grass = pi_grass / sum(pi_grass);
117     p = randperm(n_grass, C);
118     mu_grass = zeros(C, 64);
119     for i = 1 : C
120         mu_grass(i, :) = TrainsampleDCT_BG(p(i), 64);
121     end
122     sigma_grass = zeros(64, 64, C);
123     for i = 1 : C
124         randomVec = rand(1, 64);
125         sigma_grass(:, :, i) = diag(randomVec);
126     end
127     EMLimit = 1000;
128     jointPro_grass = zeros(n_grass, C);
129     for i = 1 : EMLimit
130         for j = 1 : C
131             jointPro_grass(:, j) = mvnpdf(TrainsampleDCT_BG,
132                 mu_grass(j, :), sigma_grass(:, :, j)) * pi_grass(
133                     j); % mvnpdf returns the pdf of given X, mu,
134                         sigma
135         end
136         sumRow = sum(jointPro_grass, 2);
137         likelihood_grass = sum(log(sumRow));
138         hij_grass = jointPro_grass ./ sumRow;
139         pi_grass = sum(hij_grass) / n_grass;
140         mu_grass = (hij_grass' * TrainsampleDCT_BG) ./ sum(
141             hij_grass)';
142         for j = 1:C
143             sigma_grass(:, :, j) = diag(diag(((TrainsampleDCT_BG
144                 - mu_grass(j, :))' * hij_grass(:, j))' * ...
145                 (TrainsampleDCT_BG - mu_grass(j, :)) ./ sum(
146                     hij_grass(:, j), 1)) + 0.0000001));
147         end
148         if i > 1
149             if abs(likelihood_grass - likelihood_grass_previous
150                 ) < 0.001
151                 break;
152             end
153         end

```

```

146         end
147         likelihood_grass_previous = likelihood_grass;
148     end
149 end
150
151 % BDR
152 function errorRate = ComputeBDR(zigzagImg, cheetah_mask,
    mu_cheetah, sigma_cheetah, pi_cheetah, mu_grass, sigma_grass
    , pi_grass)
153     m = 248;
154     n = 263;
155     dimensionList = [1,2,4,8,16,24,32,40,48,56,64];
156     lenList = length(dimensionList);
157     errorMat = zeros(1, lenList);
158
159     for curDim = 1 : lenList
160
161         Kth = dimensionList(curDim);
162
163         maskVec = zeros(m * n, 1); % a vector of our mask,
            will resize it later
164
165         for x = 1 : length(zigzagImg)
166
167             pro_cheetah = 0;
168             pro_grass = 0;
169
170             for y = 1:size(mu_cheetah,1)
171                 pro_cheetah = pro_cheetah + mvnpdf(zigzagImg(x
                    , 1 : Kth), mu_cheetah(y, 1 : Kth),
                    sigma_cheetah(1 : Kth, 1 : Kth,y)) *
                    pi_cheetah(y);
172             end
173
174             % compute total BDR for grass
175             for y = 1:size(mu_grass,1)
176                 pro_grass = pro_grass + mvnpdf(zigzagImg(x, 1
                    : Kth), mu_grass(y, 1 : Kth), sigma_grass(1 :
                    Kth, 1 : Kth,y))*pi_grass(y);

```

```

177         end
178
179         % decide whether the pixel is cheetah or grass
180         if pro_cheetah > pro_grass
181             maskVec(x) = 1;
182         end
183     end
184
185     maskMat = Vec2Mat(maskVec);
186
187     errorMat(curDim) = Err(cheetah_mask, maskMat);
188     errorRate = errorMat ./ (m * n);
189 end
190 end
191
192 % convert a vector into a matrix
193 function mat = Vec2Mat(vec)
194     m = 255 - 7;
195     n = 270 - 7;
196     mat = zeros(m,n);
197     for x = 1:m
198         mat(x,:) = vec(((x-1)*(n)+1):x*(n))';
199     end
200 end
201
202 % compute the error rate
203 function errorRate = Err(idealMat, ansMat)
204     errorRate = 0;
205     [m, n] = size(ansMat);
206     for i = 1 : m
207         parfor j = 1 : n
208             if(ansMat(i, j) ~= idealMat(i, j))
209                 errorRate = errorRate + 1;
210             end
211         end
212     end
213 end

```

## Code for Part B

```
1 clear
2 clc
3
4 % load all the data
5 load("ZigZagVec.mat") % 1 x 64 vector
6 load("cheetahMat.txt") % our image
7 load("TrainingSamplesDCT_8_new.mat") % the training data
   sets
8 load("zigzagImg.mat") % the preprocessed image
9 load("cheetah_mask.mat") % the preprocessed mask
10
11 % dimensions for the training & the parameters for EM
12 C = 8; % number of components
13 dimensionList = [1,2,4,8,16,24,32,40,48,56,64]; % number of
   dimensions of the space
14
15 % # of samples
16 [n_cheetah, ~] = size(TrainsampleDCT_FG);
17 [n_grass, ~] = size(TrainsampleDCT_BG);
18 totalSamples = (n_cheetah + n_grass);
19
20 % get the size of the image
21 [m, n] = size(cheetahMat);
22
23 % create a matrix
24 m = m - 7;
25 n = n - 7;
26 totalPixels = m * n;
27
28 % — EM for cheetah —
29
30 % initialize pi by creating a random normalized 1 x C matrix
31 pi_cheetah = randi(1, C);
32 pi_grass = randi(1, C);
33 pi_cheetah = pi_cheetah / sum(pi_cheetah);
34 pi_grass = pi_grass / sum(pi_grass);
35
```

```

36 % initialize mu by choosing C samples randomly from the
    training data set
37 p = randperm(n_cheetah, C); % pick C random numbers from 0
    to the size of the sample of cheetah
38 mu_cheetah = zeros(C, 64); % create an array for mu
39 mu_grass = zeros(C, 64); % create an array for mu
40 for i = 1 : C
41     mu_cheetah(i, :) = TrainsampleDCT_FG(p(i), 64);
42     mu_grass(i, :) = TrainsampleDCT_BG(p(i), 64);
43 end
44
45 % initialize sigma by creating a diagonal matrix with random
    variable
46 sigma_cheetah = zeros(64, 64, C);
47 sigma_grass = zeros(64, 64, C);
48 for i = 1 : C
49     randomVec = rand(1, 64); % generate a random vector (1
        x 64)
50     sigma_cheetah(:, :, i) = diag(randomVec); % diagonalize
        the random vector
51     randomVec = rand(1, 64); % generate a random vector (1
        x 64)
52     sigma_grass(:, :, i) = diag(randomVec); % diagonalize the
        random vector
53 end
54
55 EMLimit = 1000; % maximum iterations
56
57 % — Start of EM for cheetah —
58 jointPro_cheetah = zeros(n_cheetah, C);
59 for i = 1 : EMLimit
60     % E step
61     for j = 1 : C
62         %  $P_{\{X, Z\}}(x, z; \psi) = P_{\{X|Z\}}(x|z; \psi) * P_Z(z; \psi)$ 
        )
63         jointPro_cheetah(:, j) = mvnpdf(TrainsampleDCT_FG,
            mu_cheetah(j, :), sigma_cheetah(:, :, j)) *
            pi_cheetah(j); % mvnpdf returns the pdf of given X,
            mu, sigma

```

```

64     end
65
66     % calculate the log likelihood of the
67     sumRow = sum(jointPro_cheetah, 2); % returns the sum of
        the elements in each row
68     likelihood_cheetah = sum(log(sumRow));
69
70     % h_ij = (G(x_i, u_j, sigma_j) * pi_j) / sum(G(x_i, u_j,
        sigma_j) * pi_j)
71     hij_cheetah = jointPro_cheetah ./ sumRow;
72
73     % M step
74
75     % update pi for (n+1)
76     % pi = 1 / n * (sum(hij))
77     pi_cheetah = sum(hij_cheetah) / n_cheetah;
78
79     % update mu for (n+1)
80     % mu = sum(hij * x)/sum(hij)
81     % TODO: mu_cheetah changes dimension
82     mu_cheetah = (hij_cheetah' * TrainsampleDCT_FG) ./ sum(
        hij_cheetah)';
83
84     % update sigma for (n+1)
85     % sigma^2 = sum(hij * (x - mu)^2) / sum(hij)
86     for j = 1:C
87         sigma_cheetah(:, :, j) = diag(diag(((TrainsampleDCT_FG -
            mu_cheetah(j, :))' .* hij_cheetah(:, j))' * ...
            (TrainsampleDCT_FG - mu_cheetah(j, :)) ./ sum(
            hij_cheetah(:, j), 1)) + 0.0000001));
88
89     end
90
91     % break condition
92     % breaks the loop if the likelihood does not change more
        than 0.001
93     if i > 1
94         if abs(likelihood_cheetah -
            likelihood_cheetah_previous) < 0.001
95             break;

```

```

96         end
97     end
98
99     % store the current result as the previous
100     likelihood_cheetah_previous = likelihood_cheetah;
101
102 end
103 % —— End of EM for cheetah ——
104
105 % —— Start of EM for grass ——
106 jointPro = zeros(n_grass, C);
107 for i = 1 : EMLimit
108     % E step
109     for j = 1 : C
110         %  $P_{\{X, Z\}}(x, z; \psi) = P_{\{X|Z\}}(x|z; \psi) * P_Z(z; \psi)$ 
111         jointPro(:, j) = mvnpdf(TrainsampleDCT_BG, mu_grass(j,
            :), sigma_grass(:, :, j)) * pi_grass(j); % mvnpdf
            returns the pdf of given X, mu, sigma
112     end
113
114     % calculate the log likelihood of the
115     sumRow = sum(jointPro, 2); % returns the sum of the
            elements in each row
116     likelihood_grass = sum(log(sumRow));
117
118     %  $h_{ij} = (G(x_i, u_j, \sigma_j) * \pi_j) / \sum(G(x_i, u_j, \sigma_j) * \pi_j)$ 
119     hij_grass = jointPro ./ sumRow;
120
121     % M step
122
123     % update pi for (n+1)
124     %  $\pi = 1 / n * (\sum(h_{ij}))$ 
125     pi_grass = sum(hij_grass) / n_grass;
126
127     % update mu for (n+1)
128     %  $\mu = \sum(h_{ij} * x) / \sum(h_{ij})$ 
129     mu_grass = hij_grass' * TrainsampleDCT_BG ./ sum(hij_grass

```

```

        )';
130
131 % update sigma for (n+1)
132 % sigma^2 = sum(hij * (x - mu)^2) / sum(hij)
133 for j = 1:C
134     sigma_grass(:, :, j) = diag(diag(((TrainsampleDCT_BG -
        mu_grass(j, :))' .* hij_grass(:, j))' * ...
135     (TrainsampleDCT_BG - mu_grass(j, :)) ./ sum(
        hij_grass(:, j), 1)) + 0.0000001));
136 end
137
138 % break condition
139 % breaks the loop if the likelihood does not change more
    than 0.001
140 if i > 1
141     if abs(likelihood_grass - likelihood_grass_previous) <
        0.001
142         break;
143     end
144 end
145
146 % store the current result as the previous
147 likelihood_grass_previous = likelihood_grass;
148
149 end
150 % —— End of EM for grass ——
151
152 % —— BDR ——
153 lenList = length(dimensionList);
154 errorMat = zeros(1, lenList);
155 for curDim = 1 : lenList
156
157     Kth = dimensionList(curDim);
158
159     % compare BDR for EM
160     maskVec = zeros(m * n, 1); % a vector of our mask, will
        resize it later
161
162     for x = 1 : length(zigzagImg)

```



```

163
164     % set the probability of each class as zero
165     pro_cheetah = 0;
166     pro_grass   = 0;
167
168     % compute total BDR for cheetah
169     for y = 1:size(mu_cheetah,1)
170         pro_cheetah = pro_cheetah + mvnpdf(zigzagImg(x, 1
            : Kth), mu_cheetah(y, 1 : Kth), sigma_cheetah(1
            : Kth, 1 : Kth,y)) * pi_cheetah(y);
171     end
172
173     % compute total BDR for grass
174     for y = 1:size(mu_grass,1)
175         pro_grass = pro_grass + mvnpdf(zigzagImg(x, 1 :
            Kth), mu_grass(y, 1 : Kth), sigma_grass(1 : Kth,
            1 : Kth,y))*pi_grass(y);
176     end
177
178     % decide whether the pixel is cheetah or grass
179     if pro_cheetah > pro_grass
180         maskVec(x) = 1;
181     end
182 end
183
184 % resize the vector to matrix
185 maskMat = Vec2Mat(maskVec);
186
187 % compute the error rate
188 errorMat(curDim) = Err(cheetah_mask, maskMat);
189 errorRate = errorMat ./ (m * n);
190 end
191
192 % ——— functions ———
193
194 % convert the vector into the matrix
195 function mat = Vec2Mat(vec)
196     m = 255 - 7;
197     n = 270 - 7;

```

```

198     mat = zeros(m,n);
199     for x = 1:m
200         mat(x,:) = vec(((x-1)*(n)+1):x*(n))';
201     end
202 end
203
204 % compute the error rate
205 function errorRate = Err(idealMat, ansMat)
206     errorRate = 0;
207     [m, n] = size(ansMat);
208     for i = 1 : m
209         parfor j = 1 : n
210             if(ansMat(i, j) ~= idealMat(i, j))
211                 errorRate = errorRate + 1;
212             end
213         end
214     end
215 end

```