

1.Computation Equation

2.Curves of Classification Error vs α

3.Observation and Explanation

a)

Appendix

The following is the Matlab code.

0.1 HW5_solution.m

```
1 %%
2 clear;
3 clc;
4 tic;
5 %%
6 %Training
7 %Read the TrainingSamplesDCT_8.mat file
8 load('dataset/TrainingSamplesDCT_8_new.mat');
9 %Save TrainsampleDCT_BG and TrainsampleDCT_FG in temporary value
10 train_BG = TrainsampleDCT_BG;
11 train_FG = TrainsampleDCT_FG;
12
13 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
14 % Compute parameter from EM
15 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16 C = 8;
17 for M=1:5
18     fun_EM(C, train_BG, train_FG, M);
19 end
20 for C=[1,2,4,8,16,32]
21     fun_EM(C, train_BG, train_FG, 0);
22 end
23 %%
24 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
25 % BDR with different mixture pair and
26 % dimension
27 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28 clear all
29 %Read the TrainingSamplesDCT_8.mat file
30 load('dataset/TrainingSamplesDCT_8_new.mat');
```

```

31 %Save TrainsampleDCT_BG and TrainsampleDCT_FG in temporary value
32 train_BG = TrainsampleDCT_BG;
33 train_FG = TrainsampleDCT_FG;
34 %Compute threshold
35 P_BG = size(train_BG,1) / (size(train_BG,1) + size(train_FG,1));
36 P_FG = size(train_FG,1) / (size(train_BG,1) + size(train_FG,1));
37 T = P_BG / P_FG;
38 for FG=1:5
39     load(['savedata/MuSigma_C=8M=',num2str(FG),'.mat'],'mu_FG','sigma_FG',
40         'weight_FG');
41     for BG=1:5
42         disp(['Start with FG=',num2str(FG),',BG=',num2str(BG)]);
43         load(['savedata/MuSigma_C=8M=',num2str(BG),'.mat'],'mu_BG','sigma_BG',
44             'weight_BG');
45         error = fun_BDR(T, weight_BG, weight_FG, mu_BG, mu_FG, sigma_BG,
46             sigma_FG);
47         save(['savedata/error_FG=',num2str(FG), '&BG=',num2str(BG),'.mat'],
48             'error');
49     end
50 end
51 for C=[1,2,4,8,16,32]
52     disp(['Start with C=',num2str(C)]);
53     load(['savedata/MuSigma_C=',num2str(C), 'M=',num2str(0),'.mat'],'mu_BG',
54         ...,
55         'mu_FG','sigma_BG','sigma_FG','weight_BG','weight_FG');
56     error = fun_BDR(T, weight_BG, weight_FG, mu_BG, mu_FG, sigma_BG,
57         sigma_FG);
58     save(['savedata/error_C=',num2str(C), '&M=',num2str(0),'.mat'],'error')
59 ;
60 end
61 toc;

```

0.2 HW5_plot.m

```

1 clear all
2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 % Plot for problem a
4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 dimension = [1,2,4,8,16,24,32,40,48,56,64];
6 for FG=1:1:5
7     for BG=1:1:5
8         load(['savedata/error_FG=',num2str(FG), '&BG=',num2str(BG),'.mat'])
9         ;
10        %Plot the data
11        figure(1);
12        plot(dimension,error,'DisplayName',[ 'BG=',num2str(BG)]);
13        hold on;
14    end
15    legend

```

```

15     xlim([0 64]);
16     title(['FG=', num2str(FG)], 'FontSize', 12, 'interpreter', 'latex');
17     ylabel('PoE', 'interpreter', 'latex');
18     xlabel('Number of Dimension', 'interpreter', 'latex');
19     saveas(gcf, ['images/FG=', num2str(FG), '.jpg']);
20     close(gcf);
21 end
22
23 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
24 % Plot for problem a
25 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26 dimension = [1,2,4,8,16,24,32,40,48,56,64];
27 for C=[1,2,4,8,16,32]
28     load(['savedata/error_C=', num2str(C), '&M=', num2str(0), '.mat']);
29     %Plot the data
30     figure(1);
31     plot(dimension, error, 'DisplayName', ['Mixtures with C=', num2str(C)]);
32     hold on;
33 end
34 legend
35 xlim([0 64]);
36 title(['Mixtures with Different C'], 'FontSize', 12, 'interpreter', 'latex')
37 ;
38 ylabel('PoE', 'interpreter', 'latex');
39 xlabel('Number of Dimension', 'interpreter', 'latex');
40 saveas(gcf, ['images/mix_C.jpg']);
41 close(gcf);

```

0.3 fun_getinit.m

This function performs parameters initialization.

```

1 function [weight, mu, sigma] = fun_getinit(C,data)
2 % This function is parameter initialization
3 % C -- number of mixtures we want to use
4 %
5 % Return:
6 % weight -- the matrix of weight, size of (C,1)
7 % mu -- the matrix of mean, size of (C, 64)
8 % sigma -- tensor of sigma, size of (C, 64)
9
10 % Initialize weight
11 weight = rand(C,1);
12 weight = weight / sum(weight);
13 % Initialize mu
14 mu = 0.1 * rand(C,64) + mean(data);
15 % Initialize sigma
16 sigma = rand(C, 64);
17
18 end

```

0.4 fun_BDR.m

This function performs BDR classification using parameters we learn from EM.

```

1 function [error] = fun_BDR(T, weight_BG, weight_FG, mu_BG, mu_FG, sigma_BG
  , sigma_FG)
2 % This function is for BDR
3 % weight -- the matrix of weight, size of (C,1)
4 % mu -- the matrix of mean, size of (C, 64)
5 % sigma -- tensor of sigma, size of (C, 64)
6
7 %Load DCT file
8 load('DCT_coeffience.mat');
9 dimension = [1,2,4,8,16,24,32,40,48,56,64];
10 C = length(weight_BG);
11 error = [];
12 for d=dimension
13     DCT_coeffience_selected = DCT_coeffience(:, 1:d);
14     mu_FG_selected = mu_FG(:, 1:d);
15     mu_BG_selected = mu_BG(:, 1:d);
16     sigma_FG_selected = reshape(repmat(eye(d), [1, C])*diag(reshape(
sigma_FG(:, 1:d)', [C*d, 1])), [d, d, C]);
17     sigma_BG_selected = reshape(repmat(eye(d), [1, C])*diag(reshape(
sigma_BG(:, 1:d)', [C*d, 1])), [d, d, C]);
18     gm_FG = gmdistribution(mu_FG_selected, sigma_FG_selected, weight_FG);
19     gm_BG = gmdistribution(mu_BG_selected, sigma_BG_selected, weight_BG);
20     I_pre = pdf(gm_FG, DCT_coeffience_selected) > T * pdf(gm_BG,
DCT_coeffience_selected);
21     error_temp = fun_error(I_pre);
22     error = [error, error_temp];
23 end
24
25 end

```

0.5 fun_EM.m

This function performs EM.

```

1 function [] = fun_EM(C, data_BG, data_FG, M)
2 % This function is for EM
3 % C -- number of mixtures we want to use
4 % M -- number of mixture pair, used as a lable
5 % data_BG -- training data of BG
6 % data_FG -- training data of FG
7 %
8 % The following is abstract process:
9 % 1. Initialize parameters of mixture gaussian and weight
10 % 2. Compute initial h matrix
11 % 3. Loop of EM
12

```

```

13 loop = 1200;
14 % For BG EM
15 disp(['Start BG EM with C=',num2str(C),'M=',num2str(M)]);
16 N = size(data_BG,1); % number of samples
17 D = 64; % number of dimension
18 h_BG = zeros(N,C); % define H matrix
19 [weight_BG, mu_BG, sigma_BG] = fun_getinit(C,data_BG);
20 % weight -- the matrix of weight, size of (C,1)
21 % mu -- the matrix of mean, size of (C, 64)
22 % sigma -- tensor of sigma, size of (C, 64)
23 % h_BG -- size of (N, C)
24 % data_BG -- size of (N, 64)
25 weight_BG_pre = weight_BG;
26 mu_BG_pre = mu_BG;
27 sigma_BG_pre = sigma_BG;
28
29 for step=1:loop
30     % Compute h_BG matrix
31     for j=1:C
32         h_BG(:,j) = weight_BG_pre(j)*mvnpdf(data_BG, mu_BG_pre(j,:), diag(
sigma_BG_pre(j,:)));
33     end
34     h_BG = h_BG ./ sum(h_BG, 2);
35     % Update parameter
36     % Update weight
37     weight_BG = mean(h_BG, 1)';
38     % Update Sigma
39     sigma_temp = sum(reshape(h_BG, [1,N,C]).*(data_BG'-reshape(mu_BG_pre',
[D,1,C])).^2, 2);
40     sigma_BG = squeeze(sigma_temp)' ./ sum(h_BG, 1)';
41     sigma_BG(sigma_BG < 1e-4) = 1e-4;
42     % Update mu
43     mu_BG = h_BG'*data_BG ./ sum(h_BG, 1)';
44     % if max(max(abs(mu_BG - mu_BG_pre)./abs(mu_BG_pre))) < 0.001
45     %     break;
46     % end
47     weight_BG_pre = weight_BG;
48     mu_BG_pre = mu_BG;
49     sigma_BG_pre = sigma_BG;
50 end
51
52
53 % For FG EM
54 disp(['Start FG EM with C=',num2str(C)]);
55 N = size(data_FG,1); % number of samples
56 h_FG = zeros(N,C);
57 [weight_FG, mu_FG, sigma_FG] = fun_getinit(C,data_FG);
58 % weight -- the matrix of weight, size of (C,1)
59 % mu -- the matrix of mean, size of (C, 64)
60 % sigma -- tensor of sigma, size of (64, 64, C)

```

```

61 % h_BG -- size of (N,C)
62 % data_BG -- size of (N,64)
63 weight_FG_pre = weight_FG;
64 mu_FG_pre = mu_FG;
65 sigma_FG_pre = sigma_FG;
66
67 for step=1:loop
68     % Compute h_FG matrix
69     for j=1:C
70         h_FG(:,j) = weight_FG_pre(j)*mvnpdf(data_FG, mu_FG_pre(j,:), diag(
sigma_FG_pre(j,:)));
71     end
72     h_FG = h_FG ./ sum(h_FG, 2);
73     % Update parameter
74     % Update weight
75     weight_FG = mean(h_FG, 1)';
76     % Update sigma
77     sigma_temp = sum(reshape(h_FG, [1,N,C]).*(data_FG'-reshape(mu_FG_pre',
[D,1,C])).^2, 2);
78     sigma_FG = squeeze(sigma_temp)' ./ sum(h_FG, 1)';
79     sigma_FG(sigma_FG < 1e-4) = 1e-4;
80     % Update mu
81     mu_FG = h_FG'*data_FG ./ sum(h_FG, 1)';
82     % if max(max(abs(mu_FG - mu_FG_pre)./abs(mu_FG_pre))) < 0.001
83     % break;
84     % end
85     weight_FG_pre = weight_FG;
86     mu_FG_pre = mu_FG;
87     sigma_FG_pre = sigma_FG;
88 end
89
90 save(['savedata/MuSigma_C=', num2str(C), 'M=', num2str(M), '.mat'], 'mu_BG', ...
'mu_FG', 'sigma_BG', 'sigma_FG', 'weight_BG', 'weight_FG')
91
92 end

```

0.6 fun_error.m

This function performs error computation.

```

1 function [result] = fun_error(I_pre)
2 % This function is for computing error
3
4 %Read the mask file
5 I = imread('dataset/cheetah_mask.bmp');
6 I = im2double(I);
7 %Define the predict mask
8 mask_64 = zeros(size(I));
9 %Load DCT file
10 load('DCT_coeffience.mat');

```

```
11 %Define the loop numbers
12 loop_row = size(I,1) - 8 + 1;
13 loop_column = size(I,2) - 8 + 1;
14 k=1;
15 for i=1:1:loop_row
16     for j=1:1:loop_column
17         mask_64(i,j) = I_pre(k);
18         k=k+1;
19     end
20 end
21 %Calculate the probability of error
22 result = length(find((mask_64-I)~=0)) / (size(I,1) * size(I,2));
23 end
```