

Computer Problem Solution

a) sdf

Appendix

The following is the Matlab code.

0.1 HW3_solution.m

```
1 clear all
2 %Training
3 %Read the TrainingSamplesDCT_8.mat file
4 load('dataset/TrainingSamplesDCT_subsets_8.mat');
5 load('dataset/Alpha.mat');
6
7 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8 % Strategy 1
9 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
10 %Use strategy 1 and train set D1
11 %Save D1_BG and D1_FG in temporary value
12 train_BG = D1_BG;
13 train_FG = D1_FG;
14 fun_general(1,train_BG,train_FG,alpha,1);
15 %Use strategy 1 and train set D2
16 %Save D1_BG and D1_FG in temporary value
17 train_BG = D2_BG;
18 train_FG = D2_FG;
19 fun_general(2,train_BG,train_FG,alpha,1);
20 %Use strategy 1 and train set D3
21 %Save D1_BG and D1_FG in temporary value
22 train_BG = D3_BG;
23 train_FG = D3_FG;
24 fun_general(3,train_BG,train_FG,alpha,1);
25 %Use strategy 1 and train set D4
26 %Save D1_BG and D1_FG in temporary value
27 train_BG = D4_BG;
28 train_FG = D4_FG;
29 fun_general(4,train_BG,train_FG,alpha,1);
30
31 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
32 % Strategy 2
33 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
34 %Use strategy 2 and train set D1
35 %Save D1_BG and D1_FG in temporary value
36 train_BG = D1_BG;
37 train_FG = D1_FG;
```

```

38 fun_general(1,train_BG,train_FG,alpha,2);
39 %Use strategy 1 and train set D2
40 %Save D1_BG and D1_FG in temporary value
41 train_BG = D2_BG;
42 train_FG = D2_FG;
43 fun_general(2,train_BG,train_FG,alpha,2);
44 %Use strategy 1 and train set D3
45 %Save D1_BG and D1_FG in temporary value
46 train_BG = D3_BG;
47 train_FG = D3_FG;
48 fun_general(3,train_BG,train_FG,alpha,2);
49 %Use strategy 1 and train set D4
50 %Save D1_BG and D1_FG in temporary value
51 train_BG = D4_BG;
52 train_FG = D4_FG;
53 fun_general(4,train_BG,train_FG,alpha,2);

```

0.2 fun_general.m

This function receives parameters: train data, array of alpha and strategy we want to choose, and compute error using MAP-BDR, Bayes-BDR and ML-BDR.

```

1 function []=fun_general(dataset,train_BG,train_FG,alpha,i)
2 %This function is comparasion of using different strategy, different
   dataset and
3 %different method plugging in BDR
4 %i - represent the strategy we use
5
6 %Define the strategy we use
7 s=i; k=1;
8 %Define the array to store error
9 error_pre = zeros(size(alpha));
10 error_ml = zeros(size(alpha));
11 error_map = zeros(size(alpha));
12
13 % for Alpha=alpha
14 %     error_map(1,k) = fun_mapBDR(train_BG,train_FG,Alpha,s)
15 %     k=k+1;
16 % end
17
18 for Alpha=alpha
19     error_pre(1,k) = fun_bayesBDR(train_BG,train_FG,Alpha,s)
20     error_map(1,k) = fun_mapBDR(train_BG,train_FG,Alpha,s)
21     k=k+1;
22 end
23 error_ml(1,:)=fun_mlBDR(train_BG,train_FG);
24 %Plot the data
25 figure(1);
26 line_pre = plot(alpha,error_pre);
27 hold on;
28 line_map = plot(alpha,error_map);

```

```

29 hold on;
30 line_m1 = plot(alpha,error_m1);
31 legend({'Predictive','MAP','ML'},'Location','southeast');
32 set(gca,'XScale','log');
33 grid on;
34 title(['Dataset_',num2str(dataset),' and Strategy_',num2str(i)]...
35       ;['PoE vs Alpha']},'FontSize',12,'interpreter','latex');
36 ylabel('PoE','interpreter','latex');
37 xlabel('Alpha','interpreter','latex');
38 set(gcf,'Position',[400,100,900,600]);
39 saveas(gcf,['images/PoE of D',num2str(dataset),' and strategy'...
40           , num2str(i),'.jpg']);
41 close(gcf);
42
43 end

```

0.3 fun_bayesBDR.m

This function performs Bayes-BDR.

```

1 function [error] = fun_bayesBDR(train_BG,train_FG,Alpha,i)
2 %This function is for Bayes BDR.
3 %Use Bayes Estimation and BDR, return the predict mask of original image.
4 %i - represent the strategy we use
5
6 %Load data
7 load(['dataset/Prior_',num2str(i),'.mat']);
8 %Read the mask file
9 I = imread('dataset/cheetah_mask.bmp');
10 I = im2double(I);
11 %Define the predict mask
12 mask_64 = zeros(size(I));
13
14 P_BG = size(train_BG,1) / (size(train_BG,1) + size(train_FG,1));
15 P_FG = size(train_FG,1) / (size(train_BG,1) + size(train_FG,1));
16 %Calculate the mean of every features when cheetah
17 %mean_ch is the mean
18 mean_ch = fun_mean(train_FG);
19 %Calculate the mean of every features when grass
20 %mean_gr is the mean
21 mean_gr = fun_mean(train_BG);
22 %Compute the covariance matrix of class-condition
23 cov_ch = fun_cov(train_FG,mean_ch);
24 cov_gr = fun_cov(train_BG,mean_gr);
25 %Compute the covariance matrix of Gaussian prior
26 v = Alpha * W0;
27 cov_prior = diag(v);
28 % size(cov_prior)
29 % size(cov_ch)

```

```

30 % size(cov_gr)
31 % size(mu0_BG)
32
33 %Compute Bayes Estimation and parameters of the predictive distribution
34 %Use mu_p and mu_cov as the mean and covariance matrix of the predictive
35 %distribution. The equation is from DHS.
36 %BG predictive distribution
37 mu_p_BG = cov_prior * inv(cov_prior + cov_gr/size(train_BG,1)) * ...
38     mean_gr' + cov_gr/size(train_BG,1) * inv(cov_prior + cov_gr/size(
39         train_BG,1))...
39     * mu0_BG';
40 cov_p_BG = cov_gr + cov_prior * inv(cov_prior + cov_gr/size(train_BG,1)) *
41     cov_gr/size(train_BG,1);
42 %FG predictive distribution
43 mu_p_FG = cov_prior * inv(cov_prior + cov_ch/size(train_FG,1)) * ...
44     mean_ch' + cov_ch/size(train_FG,1) * inv(cov_prior + cov_ch/size(
45         train_FG,1))...
45     * mu0_FG';
46 cov_p_FG = cov_ch + cov_prior * inv(cov_prior + cov_ch/size(train_FG,1)) *
47     cov_ch/size(train_FG,1);
48
49 %Load DCT file
50 load('DCT_coeffience.mat');
51 %Caculate the threshold
52 T = P_BG / P_FG;
53 %Define the loop numbers
54 loop_row = size(I,1) - 8 + 1;
55 loop_column = size(I,2) - 8 + 1;
56 k=1;
57
58 for i=1:1:loop_row
59     for j=1:1:loop_column
60         P_x_FG = fun_mvgaussian(DCT_coeffience(k,:),mu_p_FG',cov_p_FG);
61         P_x_BG = fun_mvgaussian(DCT_coeffience(k,:),mu_p_BG',cov_p_BG);
62         if P_x_FG/P_x_BG > T
63             mask_64(i,j) = 1;
64         end
65     end
66     k=k+1;
67 end
68
69 %Calculate the probability of error
70 error = length(find((mask_64-I)~=0)) / (size(I,1) * size(I,2));
71
72 end

```

0.4 fun_mapBDR.m

This function performs MAP-BDR.

```
1 function [error] = fun_mapBDR(train_BG,train_FG,Alpha,i)
2 %This function is for MAP BDR.
3 %Use Bayes Estimation and BDR, return the predict mask of original image.
4 %i - represent the strategy we use
5
6 %Load data
7 load(['dataset/Prior_',num2str(i),'.mat']);
8 %Read the mask file
9 I = imread('dataset/cheetah_mask.bmp');
10 I = im2double(I);
11 %Define the predict mask
12 mask_64 = zeros(size(I));
13
14 P_BG = size(train_BG,1) / (size(train_BG,1) + size(train_FG,1));
15 P_FG = size(train_FG,1) / (size(train_BG,1) + size(train_FG,1));
16 %Calculate the mean of every features when cheetah
17 %mean_ch is the mean
18 mean_ch = fun_mean(train_FG);
19 %Calculate the mean of every features when grass
20 %mean_gr is the mean
21 mean_gr = fun_mean(train_BG);
22 %Compute the covariance matrix of class-condition
23 cov_ch = fun_cov(train_FG,mean_ch);
24 cov_gr = fun_cov(train_BG,mean_gr);
25 %Compute the covariance matrix of Gaussian prior
26 v = Alpha * W0;
27 cov_prior = diag(v);
28
29 %Compute MAP Estimation and parameters of the predictive distribution
30 %Use mu_p and mu_cov as the mean and covariance matrix of the predictive
31 %distribution. The equation is from DHS.
32 %BG predictive distribution
33 mu_p_BG = cov_prior * inv(cov_prior + cov_gr/size(train_BG,1)) * ...
34     mean_gr' + cov_gr/size(train_BG,1) * inv(cov_prior + cov_gr/size(
35     train_BG,1))...
36     * mu0_BG';
37 cov_p_BG = cov_gr;
38 %FG predictive distribution
39 mu_p_FG = cov_prior * inv(cov_prior + cov_ch/size(train_FG,1)) * ...
40     mean_ch' + cov_ch/size(train_FG,1) * inv(cov_prior + cov_ch/size(
41     train_FG,1))...
42     * mu0_FG';
43 cov_p_FG = cov_ch;
44
45 %Load DCT file
46 load('DCT_coeffience.mat');
47 %Calculate the threshold
48 T = P_BG / P_FG;
49 %Define the loop numbers
50 loop_row = size(I,1) - 8 + 1;
```

```

49 loop_column = size(I,2) - 8 + 1;
50 k=1;
51
52 for i=1:1:loop_row
53     for j=1:1:loop_column
54         P_x_FG = fun_mvgaussian(DCT_coeffience(k,:),mu_p_FG',cov_p_FG);
55         P_x_BG = fun_mvgaussian(DCT_coeffience(k,:),mu_p_BG',cov_p_BG);
56         if P_x_FG/P_x_BG > T
57             mask_64(i,j) = 1;
58         end
59         k=k+1;
60     end
61 end
62 %Calculate the probability of error
63 error = length(find((mask_64-I)~=0)) / (size(I,1) * size(I,2));
64
65 end

```

0.5 fun_mlBDR.m

This function performs ML-BDR.

```

1 function [error] = fun_bayesBDR(train_BG,train_FG)
2 %This function is for ML BDR.
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
10 P_BG = size(train_BG,1) / (size(train_BG,1) + size(train_FG,1));
11 P_FG = size(train_FG,1) / (size(train_BG,1) + size(train_FG,1));
12 %Calculate the mean of every features when cheetah
13 %mean_ch is the mean
14 mean_ch = fun_mean(train_FG);
15 %Calculate the mean of every features when grass
16 %mean_gr is the mean
17 mean_gr = fun_mean(train_BG);
18
19 %Calculate the covariance matrix of cheetah
20 cov_ch = fun_cov(train_FG,mean_ch);
21 %Calculate the covariance matrix of grass
22 cov_gr = fun_cov(train_BG,mean_gr);
23
24 %Load DCT file
25 load('DCT_coeffience.mat');
26 %Calculate the threshold
27 T = P_BG / P_FG;

```

```

28 %Define the loop numbers
29 loop_row = size(I,1) - 8 + 1;
30 loop_column = size(I,2) - 8 + 1;
31 k=1;
32
33 for z=1:1:loop_row
34     for j=1:1:loop_column
35         P_x_FG = fun_mvgaussian(DCT_coeffience(k,:),mean_ch,cov_ch);
36         P_x_BG = fun_mvgaussian(DCT_coeffience(k,:),mean_gr,cov_gr);
37         if P_x_FG/P_x_BG > T
38             mask_64(z,j) = 1;
39         end
40         k=k+1;
41     end
42 end
43 %Calculate the probability of error
44 error = length(find((mask_64-I)~=0)) / (size(I,1) * size(I,2));
45
46 end

```

0.6 fun_zigzag.m

This function compute DCT coefficients and save it as DCT_coeffience.mat.

```

1 function [] = fun_zigzag()
2 %This function is for computing the DCT coefficients of original image.
3 %Decomposition into 8 x 8 image blocks, compute the DCT of each block, and
4 %zig-zag scan.
5
6 %Read original image
7 I = imread('dataset/cheetah.bmp');
8 I = im2double(I);
9 %Define the loop numbers
10 loop_row = size(I,1) - 8 + 1;
11 loop_column = size(I,2) - 8 + 1;
12 %Read the Zig-Zag file
13 position_ref = load('dataset/Zig-Zag Pattern.txt');
14 %Define the array for saving DCT coefficients according to Zig-Zag
15 DCT_coeffience = zeros([loop_row*loop_column,64]);
16 k=1;
17
18 for i=1:1:loop_row
19     for j=1:1:loop_column
20         block = I(i:i+7,j:j+7);
21         DCT_block = dct2(block);
22         %Map DCT_block matrix to array according Zig-Zag
23         for row=1:1:8
24             for column=1:1:8

```

```
25         DCT_coeffience(k,position_ref(row,column)+1)=DCT_block(row
    ,column);
26     end
27     end
28     k=k+1;
29 end
30 end
31 save(' ../DCT_coeffience.mat','DCT_coeffience');
32 end
```

0.7 fun_mvgaussian.m

This function performs multi-gaussian.

```
1 function [y] = fun_mvgaussian(x,mean,cov)
2 %This function is for MV Gaussian
3
4 d = size(x,2);
5 y = 1/sqrt((2*pi)^d*det(cov))*exp(-(x-mean)*cov^-1*(x-mean)'/2);
6
7 end
```

0.8 fun_cov.m

```
1 function [cov] = fun_cov(data,data_mean)
2 %This function is the maximum likelihood estimation of covariance matrix
3
4 N = size(data,1);
5 cov = (data-data_mean)'*(data-data_mean)./N;
6
7 end
```

0.9 fun_mean.m

```
1 function [mean] = fun_mean(data)
2 %This function is the maximum likelihood estimation of mean
3
4 N = size(data,1);
5 A = ones(1,N);
6 mean = A * data ./ N;
7
8 end
```