

## ECE271A HW2 Report

### Problem6

a. Last week what I did is calculating prior probabilities based on the number of samples from grass and cheetah in training sets:

$$P_Y(\text{cheetah}) = 250/(1053+250) = 0.1919$$

$$P_Y(\text{grass}) = 1053/(1053+250) = 0.8081$$

This week the problem 2 shows that the optimal solution is  $c/n$ , where  $c$  is the number of times one class is observed in all observations and  $n$  is total number of observation. So the result of prior probabilities remains the same with last week's answer.

b. The Maximum Likelihood Estimate for Gaussian Classifier is based on:

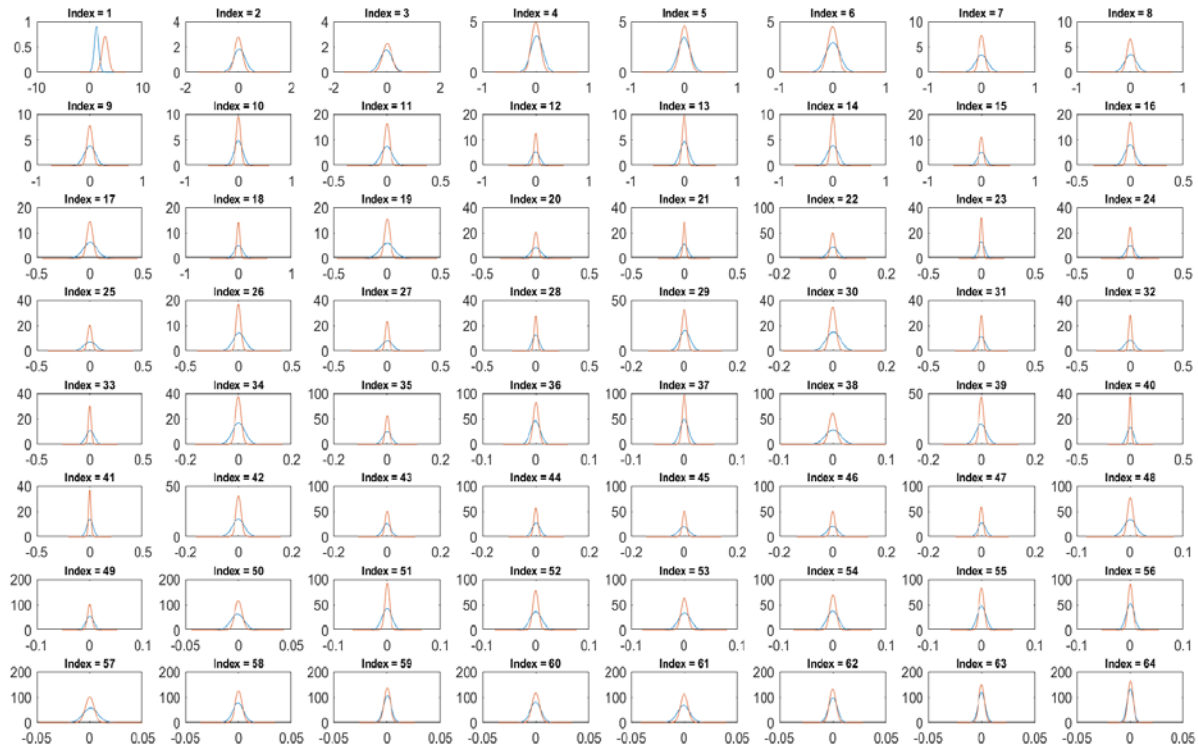
$$f(T) = \frac{1}{(\sigma_T \sqrt{2\pi})^N} e^{-\frac{1}{2} \left( \frac{T - \bar{T}}{\sigma_T} \right)^2}$$

$$L(T_1, T_2, \dots, T_N | \bar{T}, \sigma_T) = L = \prod_{i=1}^N \left[ \frac{1}{\sigma_T \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{T_i - \bar{T}}{\sigma_T} \right)^2} \right]$$

The poster class conditional probability could be calculated by:

$$P_{X|Y}(x|i) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}} \exp \left\{ -\frac{1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) \right\}$$

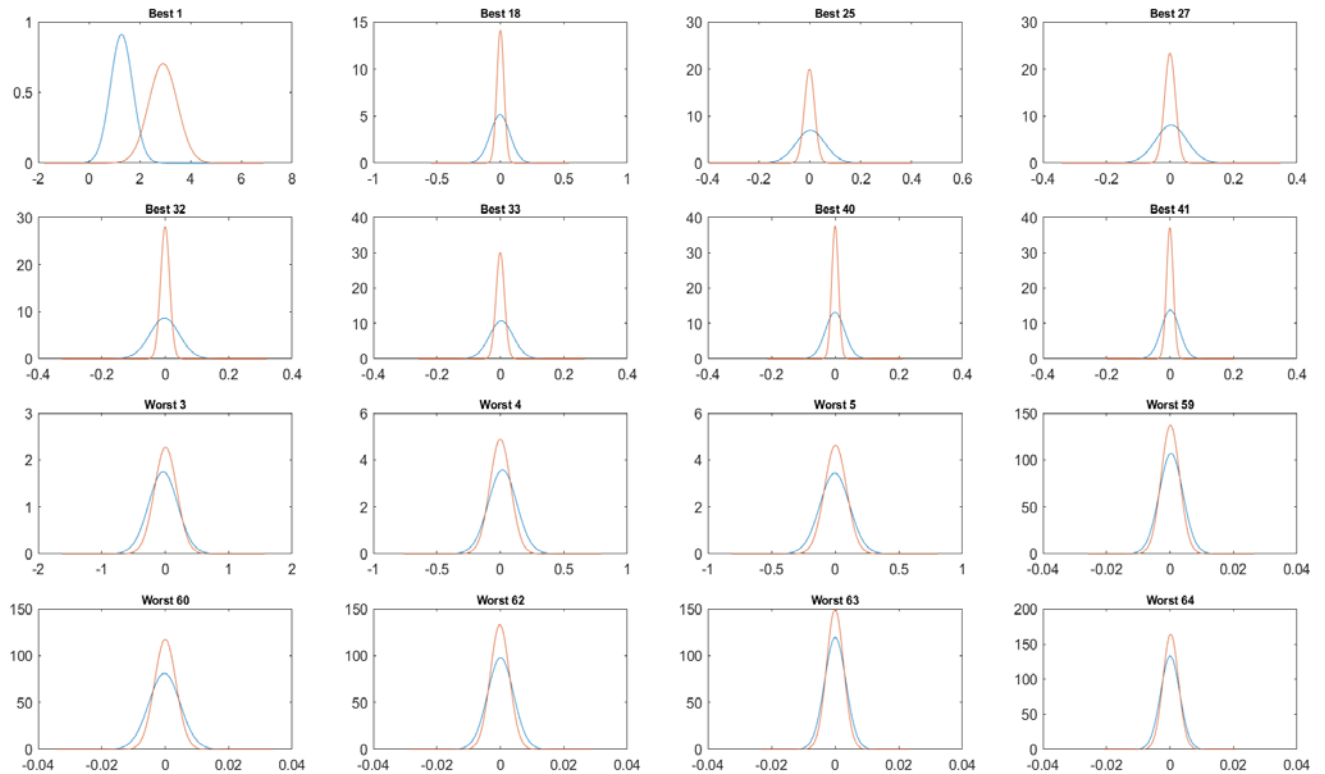
This process is included in my code. As for the marginal distribution plots, they are here:



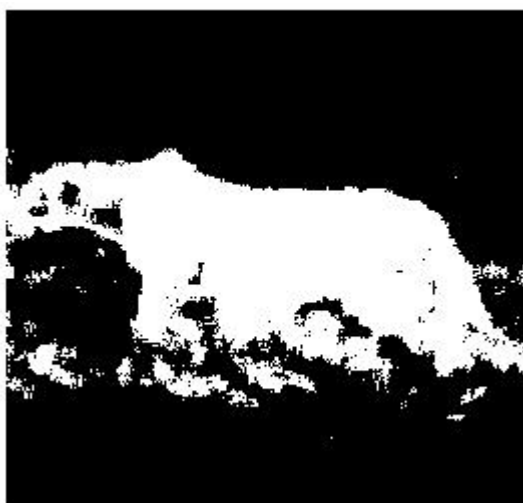
Then I used Kullback-Leibler distance (KL distance) to measure the similarity of the 2-class marginal distribution. The bigger difference indicates that the feature in this dimension has higher value for classification.

The 8 best index is [1, 18, 25, 27, 32, 33, 40, 41].

The 8 worst index is [3, 4, 5, 59, 60, 62, 63, 64].



c. The Bayesian Decision Rule is applied to given image to give prediction based on training set.



(i) 64 dimension feature prediction



(ii) 8 best feature prediction

Probability of error is calculated based on:

$$\text{error}(\text{FG}) = \frac{\text{\#FG pixels misclassified as BG}}{\text{\#FG pixels in ground truth of test set}} \times \text{prior probability of FG}$$

$$\text{error}(\text{BG}) = \frac{\text{\#BG pixels misclassified as FG}}{\text{\#BG pixels in ground truth of test set}} \times \text{prior probability of BG}$$

$$\text{error} = \text{error}(\text{FG}) + \text{error}(\text{BG})$$

The result of 64 dimension feature is  $\text{error}(\text{FG}) = 0.0107$ ,  $\text{error}(\text{BG})=0.1250$ , **error = 0.1357**

The result of 8 dimension feature is  $\text{error}(\text{FG}) = 0.0143$ ,  $\text{error}(\text{BG})=0.0334$ , **error = 0.0477**

The result shows that the performance of 8 best features we chose is better than all 64 dimension features, which means that the feature selection is significant for classification task and the best feature should be the one whose distribution on different classes are not similar. One of the possible reason would be that there exist bad features among 64 dimension features. These bad features is hard to determine whether the given sample pixel is cheetah or grass so that they will make the prediction result bad. However, the good feature does not have similar probability distribution, which means that it is easy to use them to classify whether the given pixel is cheetah or grass. In this way, using 8 best features is better than using all 64 features.

Code:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% problem a
n = 1053+250;
c_cheetah = 250;
c_grass = 1053;
prior_cheetah = c_cheetah/n;
prior_grass = c_grass/n;

% From Problem 2 we got  $p^* = c/n$ 
% where c is the number of times that a certain class samples are observed.
% n is the number of times for observation totally.
% Compared to the result of last week, the prior probability is identical.
% Last week, the result of prior probability is obtained from number of observation directly.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% problem b
train_set = load('TrainingSamplesDCT_8_new.mat');
FGmat = train_set.TrainsampleDCT_FG;
BGmat = train_set.TrainsampleDCT_BG;

%%% maximum likelihood estimates
%%% foreground
meanFG = mean(FGmat);
covFG = cov(FGmat);
varFG = var(FGmat);
mleFG = zeros(c_cheetah,1);
for i=1:c_cheetah
    tmp = 0;
    for j=1:64
        tmp = tmp + ((FGmat(i,j)-meanFG(j))/sqrt(varFG(j)))^2;
    end
    mleFG(i,1) = exp(-32*log(2*pi)-64*log(sqrt(varFG(j)))-0.5*tmp);
end
%%% background
meanBG = mean(BGmat);
covBG = cov(BGmat);
varBG = var(BGmat);
mleBG = zeros(c_grass,1);
for i=1:c_grass
    ave = mean(BGmat(i,:));
    sigma = std(BGmat(i,:));
    tmp = 0;
    for j=1:64
        tmp = tmp + ((BGmat(i,j)-meanBG(j))/sqrt(varBG(j)))^2;
    end
    mleBG(i,1) = exp(-32*log(2*pi)-64*log(sqrt(varBG(j)))-0.5*tmp);
end

%%% marginal plot
kl = [];
for i=1:64
    subplot(8,8,i)
    set(gcf,'Position',get(0,'ScreenSize'))
    ave_FG = mean(FGmat(:,i));
    variance_FG = var(FGmat(:,i));
    sigma_FG = sqrt(variance_FG);
    ave_BG = mean(BGmat(:,i));
    variance_BG = var(BGmat(:,i));
    sigma_BG = sqrt(variance_BG);
    xFG = ave_FG-7*sigma_FG:sigma_FG/50:ave_FG+7*sigma_FG;
    xBG = ave_BG-7*sigma_BG:sigma_BG/50:ave_BG+7*sigma_BG;
    x = sort([xFG xBG]);
    y_cheetah = normpdf(x,ave_FG,sigma_FG);
    y_grass = normpdf(x,ave_BG,sigma_BG);

    plot(x,y_cheetah,x,y_grass)
    title(['Index = ' num2str(i)], 'FontSize',8);
end
```

```

    % calculate KL distance
    kl_tmp = KLdiv(y_cheetah,y_grass);
    kl = [kl kl_tmp];
end
print('64_plots','-dpng');

%%% Find Best and Worst based on KL distance
[KLSorted, KLIIdx] = sort(kl);
worstidx = sort(KLIIdx(1:8));
bestidx = sort(KLIIdx(57:64));

figure();
%%% Make output for best and worst cases
for i=1:8
    idx_w = worstidx(i);
    idx_b = bestidx(i);
    % Worst
    subplot(4,4,i+8)
    ave_FG = mean(FGmat(:,idx_w));
    variance_FG = var(FGmat(:,idx_w));
    sigma_FG = sqrt(variance_FG);
    ave_BG = mean(BGmat(:,idx_w));
    variance_BG = var(BGmat(:,idx_w));
    sigma_BG = sqrt(variance_BG);
    xFG = ave_FG-7*sigma_FG:sigma_FG/50:ave_FG+7*sigma_FG;
    xBG = ave_BG-7*sigma_BG:sigma_BG/50:ave_BG+7*sigma_BG;
    x = sort([xFG xBG]);
    y_cheetah = normpdf(x,ave_FG,sigma_FG);
    y_grass = normpdf(x,ave_BG,sigma_BG);
    plot(x,y_cheetah,x,y_grass)
    title(['Worst ' num2str(idx_w)],'FontSize',8);
    % Best
    subplot(4,4,i)
    ave_FG = mean(FGmat(:,idx_b));
    variance_FG = var(FGmat(:,idx_b));
    sigma_FG = sqrt(variance_FG);
    ave_BG = mean(BGmat(:,idx_b));
    variance_BG = var(BGmat(:,idx_b));
    sigma_BG = sqrt(variance_BG);
    xFG = ave_FG-7*sigma_FG:sigma_FG/50:ave_FG+7*sigma_FG;
    xBG = ave_BG-7*sigma_BG:sigma_BG/50:ave_BG+7*sigma_BG;
    x = sort([xFG xBG]);
    y_cheetah = normpdf(x,ave_FG,sigma_FG);
    y_grass = normpdf(x,ave_BG,sigma_BG);
    plot(x,y_cheetah,x,y_grass)
    title(['Best ' num2str(idx_b)],'FontSize',8);
end
set(gcf,'Position',get(0,'ScreenSize'))
print('BestandWorst_plots','-dpng');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% problem c
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% (i) Use 64-dimension Gaussian
img = imread('cheetah.bmp');
img = im2double(img);
%%% Add paddle
img = [zeros(size(img,1),4) img];
img = [zeros(4, size(img,2)); img];
img = [img zeros(size(img,1),3)];
img = [img; zeros(3, size(img,2))];

%%% DCT
[m,n] = size(img);
Blocks = ones(m-7,n-7);
mean_FG = mean(FGmat);
mean_BG = mean(BGmat);
ave_tmp_FG = mean_FG;

```

```

ave_tmp_BG = mean_BG;
inv_covFG = inv(covFG);
inv_covBG = inv(covBG);
DcovFG = det(covFG);
DcovBG = det(covBG);

%%% predict
for i=1:m-7
    for j=1:n-7
        DCT = dct2(img(i:i+7,j:j+7));
        zigzag_order = zigzag(DCT);
        feature = zigzag_order;
        g_cheetah = 0;
        g_grass = 0;
        % cheetah
        g_cheetah = g_cheetah + feature*inv_covFG*transpose(feature);
        g_cheetah = g_cheetah - 2*feature*inv_covFG*transpose(ave_tmp_FG);
        g_cheetah = g_cheetah + ave_tmp_FG*inv_covFG*transpose(ave_tmp_FG);
        g_cheetah = g_cheetah + log(DcovFG) - 2*log(prior_cheetah);
        % grass
        g_grass = g_grass + feature*inv_covBG*transpose(feature);
        g_grass = g_grass - 2*feature*inv_covBG*transpose(ave_tmp_BG);
        g_grass = g_grass + ave_tmp_BG*inv_covBG*transpose(ave_tmp_BG);
        g_grass = g_grass + log(DcovBG) - 2*log(prior_grass);
        if g_cheetah >= g_grass
            Blocks(i,j) = 0;
        end
    end
end

%%% save prediction
imwrite(Blocks, 'prediction_64d.jpg');
prediction = mat2gray(Blocks);

ground_truth = imread('cheetah_mask.bmp')/255;
x = size(ground_truth, 1);
y = size(ground_truth, 2);
count1 = 0;
count2 = 0;
count_cheetah_truth = 0;
count_grass_truth = 0;
for i=1:x
    for j=1:y
        if prediction(i,j) > ground_truth(i,j)
            count2 = count2 + 1;
            count_grass_truth = count_grass_truth + 1;
        elseif prediction(i,j) < ground_truth(i,j)
            count1 = count1 + 1;
            count_cheetah_truth = count_cheetah_truth + 1;
        elseif ground_truth(i,j) > 0
            count_cheetah_truth = count_cheetah_truth + 1;
        else
            count_grass_truth = count_grass_truth + 1;
        end
    end
end
error1_64 = (count1/count_cheetah_truth) * prior_cheetah;
error2_64 = (count2/count_grass_truth) * prior_grass;
total_error_64 = error1_64 + error2_64;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% (ii) Use 8 best features
img = imread('cheetah.bmp');
img = im2double(img);
%%% Add paddle
img = [zeros(size(img,1),4) img];
img = [zeros(4, size(img,2)); img];
img = [img zeros(size(img,1),3)];
img = [img; zeros(3, size(img,2))];

```



```

function output = zigzag(in)
% initializing the variables
%-----
h = 1;
v = 1;
vmin = 1;
hmin = 1;
vmax = size(in, 1);
hmax = size(in, 2);
i = 1;
output = zeros(1, vmax * hmax);
%-----
while ((v <= vmax) && (h <= hmax))

    if (mod(h + v, 2) == 0)          % going up
        if (v == vmin)
            output(i) = in(v, h);    % if we got to the first line
            if (h == hmax)
                v = v + 1;
            else
                h = h + 1;
            end
            i = i + 1;
        elseif ((h == hmax) && (v < vmax)) % if we got to the last column
            output(i) = in(v, h);
            v = v + 1;
            i = i + 1;
        elseif ((v > vmin) && (h < hmax)) % all other cases
            output(i) = in(v, h);
            v = v - 1;
            h = h + 1;
            i = i + 1;
        end

    else                                % going down
        if ((v == vmax) && (h <= hmax)) % if we got to the last line
            output(i) = in(v, h);
            h = h + 1;
            i = i + 1;

        elseif (h == hmin)              % if we got to the first column
            output(i) = in(v, h);
            if (v == vmax)
                h = h + 1;
            else
                v = v + 1;
            end
            i = i + 1;
        elseif ((v < vmax) && (h > hmin)) % all other cases
            output(i) = in(v, h);
            v = v + 1;
            h = h - 1;
            i = i + 1;
        end
    end
    if ((v == vmax) && (h == hmax))      % bottom right element
        output(i) = in(v, h);
        break
    end
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function KLD=KLdiv(P,Q)
P = P/sum(P);
Q = Q/sum(Q);
index = P > 0;
kld = -1 * P(index).* log(Q(index)./P(index));
kld(isinf(kld)|isnan(kld)) = 0;
KLD = sum(kld);
end

```