Student Name: Lin-Hsi (Ethen) Tsao
Student PID: A59015911

Statistical Learning 1 (ECE 271A)

HW2

(a) **Using the training data in TrainingSamplesDCT 8.mat compute the histogram estimate of the prior $P_Y(i), i \in \{cheetah, grass\}$. Using the results of problem 2 compute the maximum likelihood estimate for the prior probabilities. Compare the result with the estimates that you obtained last week. If they are the same, interpret what you did last week. If they are different, explain the differences.**

The maximum likelihood estimate for the prior probabilities is the same as the previous estimation last week.

$$\text{size of total training sample} = \text{size of cheetah training sample} + \text{size of grass training sample}$$

$$P_Y(cheetah) = \frac{\text{size of cheetah training sample}}{\text{size of total training sample}}$$

$$P_Y(grass) = \frac{\text{size of grass training sample}}{\text{size of total training sample}}$$

In this case, $P_Y(cheetah) = 0.1919, P_Y(grass) = 0.8081$.

(b) **Using the training data in TrainingSamplesDCT 8.mat, compute the maximum likelihood estimates for the parameters of the class conditional densities $P_{(X|Y)}(x|cheetah)$ and $P_{(X|Y)}(x|grass)$ under the Gaussian assumption. Denoting by $X = \{X_1, ..., X_{64}\}$ the vector of DCT coefficients, create 64 plots with the marginal densities for the two classes - $P_{(X_k|Y)}(x_k|cheetah)$ and $P_{(X_k|Y)}(x_k|grass)$), $k = 1, ..., 64$ - on each. Use different line styles for each marginal. Select, by visual inspection, what you think are the best 8 features for classification purposes and what you think are the worst 8 features (you can use the subplot command to compare several plots at a time). Hand in the plots of the marginal densities for the best-8 and worst-8 features (once again you can use subplot, this should not require more than two sheets of paper). In each subplot indicate the feature that it refers to.**

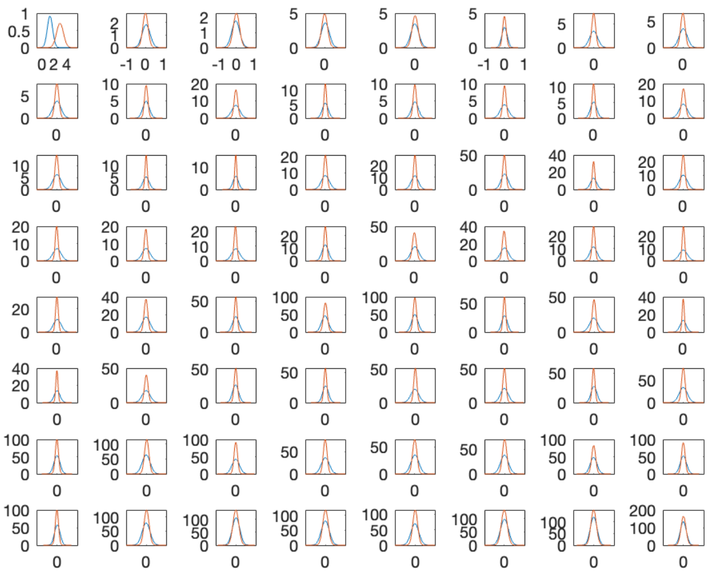
Given the marginal densities of the two classes - $P_{(X_k|Y)}(x_k|cheetah)$ and $P_{(X_k|Y)}(x_k|grass)$) are Gaussian assumption, we can plot the densities and compute the maximum likelihood by the following equation.

$$\text{estimated } mean = \text{sample } mean$$

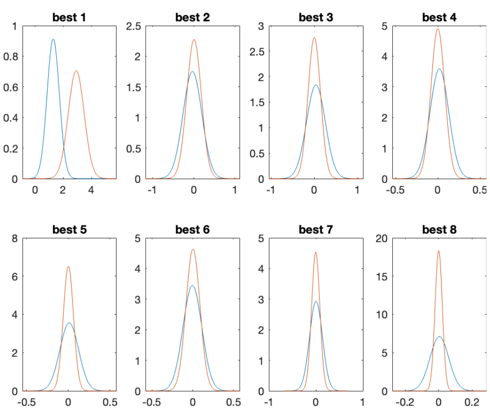
$$\text{estimated } std = \text{sample } std$$

$$L = \frac{1}{(\sigma_T\sqrt{2\pi})^N} e^{-\frac{1}{2}\sum_{i=1}^{N}\left(\frac{T_i - \bar{T}}{\sigma_T}\right)^2} \text{ In this case, N = 64.}$$

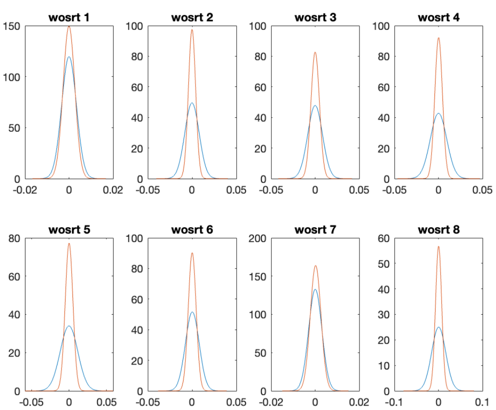
The following graphs are the densities of all the features, the best 8 features, and the worst 8 features. We can notice that the better feature is, the more distance between the mean of the foreground and the background is.



(a) All features



(b) Best 8 features



(c) Worst 8 features

(c) **Compute the Bayesian decision rule and classify the locations of the cheetah image using i) the 64-dimensional Gaussians, and ii) the 8-dimensional Gaussians associated with the best 8 features. For the two cases, plot the classification masks and compute the probability of error by comparing with cheetah mask.bmp. Can you explain the results?**

For this problem, we could use the DCT process as HW1. While determining the classification, we use the Gaussian classifier. The folloeing 5 steps are my approach.

1) To normalize the value in the range [0, 1], divide img *cheetah.bmp* by 255.
2) Padding the img *cheetah.bmp* with 0 for all four sides to get a new image size with (263, 278). After performing sliding windows, we could get the matrix with the size of (255, 270), the same as the input img, by padding.
3) For each 8*8 block, we compute DCT, order coefficients with zig-zag scan.
4) Determine the predicted class in the current block by Gaussian classifier. The BDR is

$$i^*(x) = \underset{i}{argmax}[-\frac{1}{2}(x-\mu_i)^T \sum_i^{-1}(x-\mu_i) - \frac{1}{2}log(2\pi)^d|\sum_i| + logP_Y(i)]$$

where x denotes the flatten vector from each DCT sliding blocks, while we can get $\sum_i$ and μ_i from calculation. The predicted class = 0 (background) if the value with i = 0 in the argmax function is larger than that with i = 1 in the argmax function. Likewise, The predicted class = 1 (foreground) if the value with i = 1 in the argmax function is larger than or equal to that with i = 0 in the argmax function.

5) Create a binary mask *all* with 1's for cheetah (foreground) and 0's for grass (background).
6) Repeat step 1. to step 5. with the best 8 features and create a binary mask *best*8.

(a) All features

(b) Best 8 features

For all the pixel in the binary mask *all* and *best*8, we could check whether the predicted label equals to the ground truth label in *cheetah_mask.bmp*. The error rate could be calculated by

$$\text{error rate} = \frac{\text{number of different pixel value}}{\text{number of pixels in the mask}}$$

In this case, error rate with all features = **0.1357** and error rate with best 8 features = **0.1000**. Both the values are smaller than that we obtained from HW1.

Code

HW2_1.m HW2_2.m HW2_3.m HW1_4.m

```
load('TrainingSamplesDCT_8_new.mat');
total = size(TrainsampleDCT_FG) + size(TrainsampleDCT_BG);
priorFG = length(TrainsampleDCT_FG) / total(1);
priorBG = length(TrainsampleDCT_BG) / total(1);
```

Figure 3: Code for problem (a)

HW2_1.m HW2_2.m HW2_3.m HW1_4.m

```
% Compute the MLE for the foreground;
fSize = size(TrainsampleDCT_FG);
fSize = fSize(1);
fBase = zeros(1, 64);
fTmp = zeros(1, 64);
fMean = mean(TrainsampleDCT_FG, 1);
fVar = var(TrainsampleDCT_FG, 1);
fStd = std(TrainsampleDCT_FG, 1);

for col = 1 : 64
    fBase(col) = (-2/fSize) * log (2 * pi);
end

for col = 1 : 64
    for row = 1 : fSize
        fTmp(col) = fTmp(col) + ((TrainsampleDCT_FG(row, col) - fMean(col)) / fStd(col))^2;
    end

    fTmp(col) = 0.5 * fTmp(col);
end

fMLELog = fBase - fSize * log(fStd) - fTmp;

% Compute the MLE for the background;
bSize = size(TrainsampleDCT_BG);
bSize = bSize(1);
bBase = zeros(1, 64);
bTmp = zeros(1, 64);
bMean = mean(TrainsampleDCT_BG, 1);
bVar = var(TrainsampleDCT_BG, 1);
bStd = std(TrainsampleDCT_BG, 1);

for col = 1 : 64
    bBase(col) = (-2/bSize) * log (2 * pi);
end

for col = 1 : 64
    for row = 1 : bSize
        bTmp(col) = bTmp(col) + ((TrainsampleDCT_BG(row, col) - bMean(col)) / bStd(col))^2;
    end

    bTmp(col) = 0.5 * bTmp(col);
end

bMLELog = bBase - bSize * log(bStd) - bTmp;

% plot the image;
sampleNum = 100;
stdNum = 5;
```

HW2_1.m HW2_2.m HW2_3.m HW1_4.m

```
distance = zeros(1, 64);

for i = 1 : 64
    subplot(8, 8, i);
    fX = linspace(fMean(i) - stdNum * fStd(i), fMean(i) + stdNum * fStd(i), sampleNum);
    bX = linspace(bMean(i) - stdNum * bStd(i), bMean(i) + stdNum * bStd(i), sampleNum);
    x = sort([fX, bX]);

    fY = normpdf(x, fMean(i), fStd(i));
    bY = normpdf(x, bMean(i), bStd(i));
    distance(i) = abs(fMean(i) - bMean(i));

    plot(x, fY, x, bY);
end
savefig('all features.fig');

% plot the best and the worst feature;
[out, idx] = sort(distance, "descend");
best = idx(1 : 8);
worst = fliplr(idx(57 : 64));

% best;
for i = 1 : size(best, 2)
    subplot(2, 4, i);
    fX = linspace(fMean(best(i)) - stdNum * fStd(best(i)), fMean(best(i)) + stdNum * fStd(best(i)), sampleNum);
    bX = linspace(bMean(best(i)) - stdNum * bStd(best(i)), bMean(best(i)) + stdNum * bStd(best(i)), sampleNum);
    x = sort([fX, bX]);

    fY = normpdf(x, fMean(best(i)), fStd(best(i)));
    bY = normpdf(x, bMean(best(i)), bStd(best(i)));

    plot(x, fY, x, bY);
    title(['best ', num2str(i)]);
end
savefig('best 8 features.fig');

% worst;
for i = 1 : size(worst, 2)
    subplot(2, 4, i);
    fX = linspace(fMean(worst(i)) - stdNum * fStd(worst(i)), fMean(worst(i)) + stdNum * fStd(worst(i)), sampleNum);
    bX = linspace(bMean(worst(i)) - stdNum * bStd(worst(i)), bMean(worst(i)) + stdNum * bStd(worst(i)), sampleNum);
    x = sort([fX, bX]);

    fY = normpdf(x, fMean(worst(i)), fStd(worst(i)));
    bY = normpdf(x, bMean(worst(i)), bStd(worst(i)));

    plot(x, fY, x, bY);
    title(['wosrt ', num2str(i)]);
end
```

Figure 4: Code for problem (b)

HW2_1.m HW2_2.m HW2_3.m HW1_4.m +

```
% read img;
img = imread('cheetah.bmp');
img = double(img) / 255;

% padding img with zero;
I = zeros(263, 278);
for row = 5 : 259
    for col = 5 : 274
        I(row, col) = img(row - 4, col - 4);
    end
end

% read pattern;
pattern = readmatrix('Zig-Zag Pattern.txt');

% create the bitmask by all features;
all = zeros(255, 270);

fCov = cov(TrainsampleDCT_FG);
bCov = cov(TrainsampleDCT_BG);
fCovInv = inv(fCov);
bCovInv = inv(bCov);
fCovDet = det(fCov);
bCovDet = det(bCov);
fAllMean = mean(TrainsampleDCT_FG, 1);
bAllMean = mean(TrainsampleDCT_BG, 1);

for row = 1 : 255
    for col = 1 : 270
        block = zeros(8, 8);
        % get the blcok;
        for r = row : row + 7
            for c = col : col + 7
                block(r - row + 1, c - col + 1) = I(r, c);
            end
        end

        % compute DCT;
        dct2Block = dct2(block);
        flatBlock = zeros(1, 64);

        % order coefficients with zig-zag scan;
        for r = 1 : 8
            for c = 1 : 8
                flatBlock(1, pattern(r, c) + 1) = dct2Block(r, c);
            end
        end

        % use Gaussian classifier to find class Y for each block;
```

HW2_1.m HW2_2.m HW2_3.m HW1_4.m +

```
        fRes = -0.5 * (flatBlock - fAllMean) * fCovInv * transpose(flatBlock - fAllMean) - ...
        0.5 * log((2 * pi) ^ 64 * fCovDet) + log(priorFG);

        bRes = -0.5 * (flatBlock - bAllMean) * bCovInv * transpose(flatBlock - bAllMean) - ...
        0.5 * log((2 * pi) ^ 64 * bCovDet) + log(priorBG);

        % create a binary mask;
        if fRes >= bRes
            all(row, col) = 1;
        else
            all(row, col) = 0;
        end
    end
end

imshow(uint8(all), [0 1]);
savefig('all.fig');

% create the bitmask by best 8 features;
best8 = zeros(255, 270);

fCov = cov(TrainsampleDCT_FG(:, best));
bCov = cov(TrainsampleDCT_BG(:, best));
fCovInv = inv(fCov);
bCovInv = inv(bCov);
fCovDet = det(fCov);
bCovDet = det(bCov);
fBestMean = mean(TrainsampleDCT_FG(:, best), 1);
bBestMean = mean(TrainsampleDCT_BG(:, best), 1);

for row = 1 : 255
    for col = 1 : 270
        block = zeros(8, 8);
        % get the blcok;
        for r = row : row + 7
            for c = col : col + 7
                block(r - row + 1, c - col + 1) = I(r, c);
            end
        end

        % compute DCT;
        dct2Block = dct2(block);
        flatBlock = zeros(1, 64);

        % order coefficients with zig-zag scan;
        for r = 1 : 8
            for c = 1 : 8
                flatBlock(1, pattern(r, c) + 1) = dct2Block(r, c);
            end
```

HW2_1.m HW2_2.m HW2_3.m HW1_4.m +
            for c = 1 : 8
                flatBlock(1, pattern(r, c) + 1) = dct2Block(r, c);
            end

```
        end
        flatBlock = flatBlock(best);

        % use Gaussian classifier to find class Y for each block;
        fRes = -0.5 * (flatBlock - fBestMean)  * fCovInv * transpose(flatBlock - fBestMean) - ...
        0.5 * log((2 * pi) ^ 64 * fCovDet) + log(priorFG);

        bRes = -0.5  * (flatBlock - bBestMean) * bCovInv * transpose(flatBlock - bBestMean) - ...
        0.5 * log((2 * pi) ^ 64 * bCovDet) + log(priorBG);

        % create a binary mask;
        if fRes >= bRes
            best8(row, col) = 1;
        else
            best8(row, col) = 0;
        end
    end
end

imshow(uint8(best8), [0 1]);
savefig('best8.fig');

% compute the error rate;
mask = imread('cheetah_mask.bmp');
allErrorCount = 0;
best8ErrorCount = 0;
sizes = size(mask);
rows = sizes(1);
cols = sizes(2);

% check whether the predicted label equals to the ground truth label;
for row  = 1 : rows
    for col  = 1 : cols
        if (mask(row, col) / 255 ~= all(row, col))
            allErrorCount = allErrorCount + 1;
        end
        if (mask(row, col) / 255 ~= best8(row, col))
            best8ErrorCount = best8ErrorCount + 1;
        end
    end
end

allErrorRate = allErrorCount / (rows * cols);
disp(allErrorRate);
best8ErrorRate = best8ErrorCount / (rows * cols);
disp(best8ErrorRate);
```

Figure 5: Code for problem (c)