ECE 271A - Statistical Learning
Homework Set Four

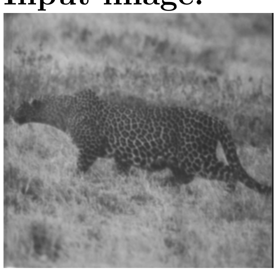
You-Yi Jau
Collaborator: Yi-Qian Wang

November 21, 2018

1 Introduction

Segmentation of Cheetah: Using Bayesian probability with multi-variant gaussian model to segment 'cheetah' from background.

Input image:

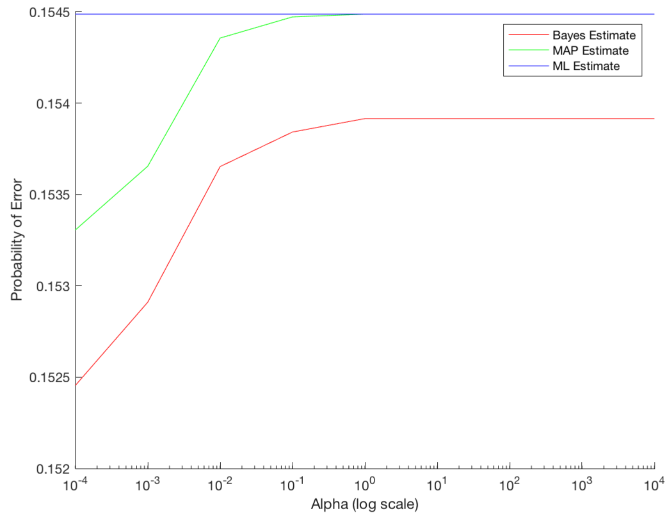


2 Problems and Results

Once again we use the de-composition into 8 8 image blocks, compute the DCT of each block, and zig-zag scan. We also continue to assume that the class-conditional densities are multivariate Gaussians of 64 dimensions. The goal is to understand the benefits of a Bayesian solution.

1. **The relative behavior of these three curves: Bayes-Bayesian Decision Rule (BDR), ML-BDR, Bayes MAP-BDR**

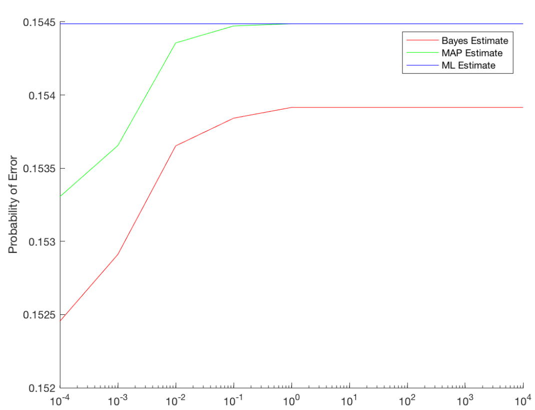
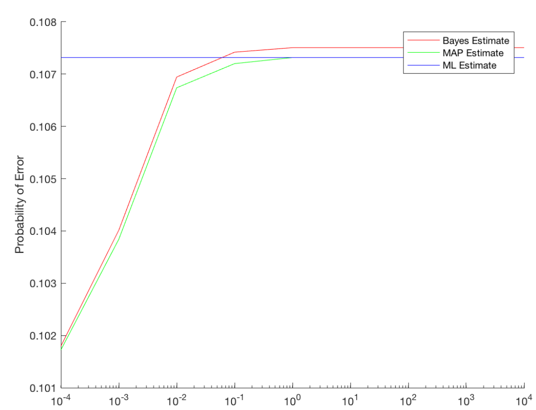
    - The curves of classification error as a function of α. (Given the training set $D1$):

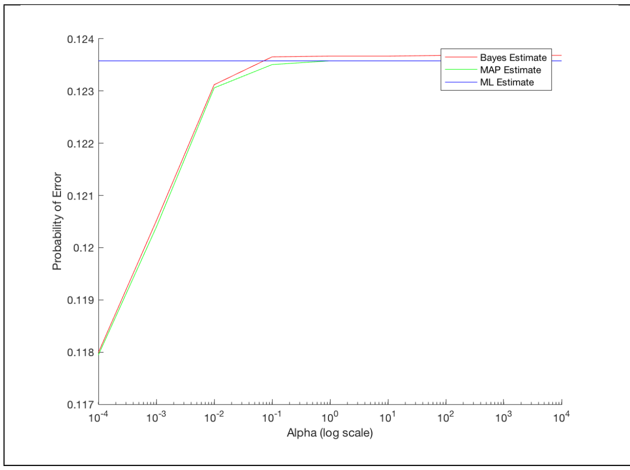
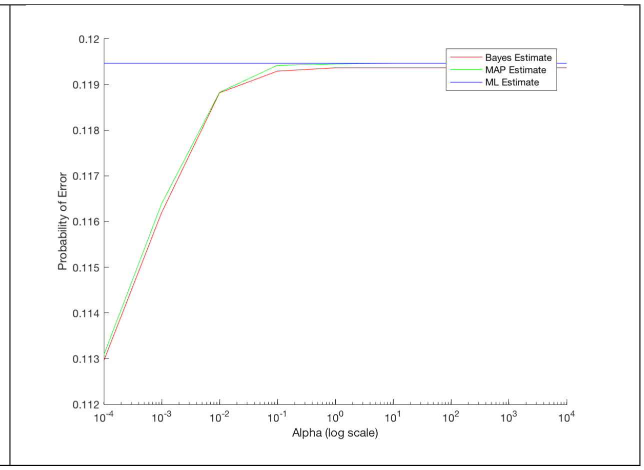


Explanation:
- As we can see, the bayes estimator has the lowest probability of error (PoE) under all alphas, as shown by the red curve. It is because the prior helps when the dataset is small.
- The MAP (green curve) is in the middle because the prior mean helps when training set is small. MAP makes us trust one parameter that maximize $P(mu|Dataset)$.
- The ML estimator doesn't be affected by the value of Alpha. It only considers the sample mean and variance.
- On the other hand, When alpha is small, the uncertainty of prior is low. This makes us trust prior more. When alpha is getting higher, we trust prior less. That's why the PoE is getting closer when alpha is large.

- Under different datasets ($D1$, $D2$, $D3$, $D4$), using strategy one:

Dataset size: (cheetah examples, grass examples).

Dataset 1 (75, 300)	Dataset 2 (125, 500)
	
Dataset 3 (175, 700)	Dataset 4 (225, 900)

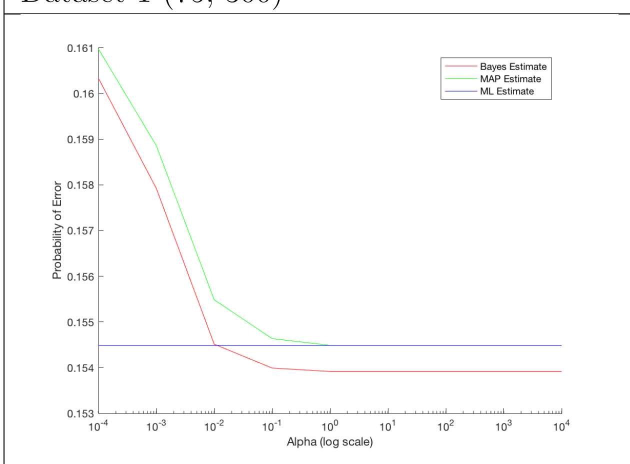
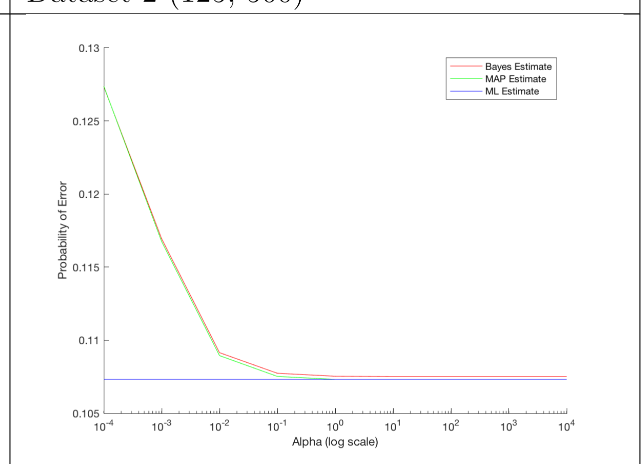
	

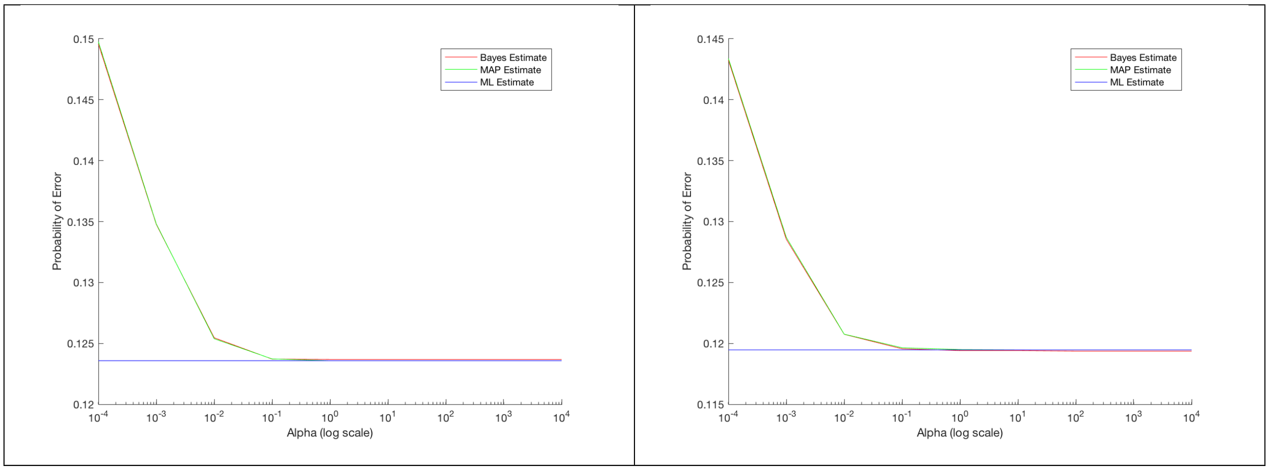
Explanation:

- The PoE of D2, D3, D4 is lower than D1.
- In D2, D3, D4, when alpha gets higher, the PoE converges to similar level.
- The prior mu helps to reduce error rate. When alpha is small, we trust prior more than the time when alpha is large.
- Larger dataset doesn't ensure better predition, as there may be some outliers in the training data.

- Under different strategies (Strategy 1, strategy 2):

Dataset size: (cheetah examples, grass examples).

Dataset 1 (75, 300)	Dataset 2 (125, 500)
	
Dataset 3 (175, 700)	Dataset 4 (225, 900)



Explanation:
- The result looks like oppisite of that in strategy 1.
- The prior is not trust-worthy because the mu1 and mu2 are identical.
- If we trust the prior knowledge, we get bad result.
- The PoE converges when alpha gets high, which means that we trust the prior less.

- Prior probabilities (Take D1 for example):

The probability when j = cheetah, and j = grass is

$$P_Y(cheetah) = \frac{size\ of\ Train\_sample\_DCT\_FG}{size\ of\ all\ training\ samples} = \frac{75}{75+300} = 0.2, \tag{1}$$

$$P_Y(grass) = \frac{size\ of\ Train\_sample\_DCT\_BG}{size\ of\ all\ training\ samples} = \frac{300}{75+300} = 0.8, \tag{2}$$

Probability of error =

$$\frac{Wrong\_prediction\_pixels\_of\_FG}{All\_pixels\_of\_FG} * P_Y(cheetah) + \frac{Wrong\_prediction\_pixels\_of\_BG}{All\_pixels\_of\_BG} * P_Y(grass) \tag{3}$$

References

[1] *Pattern recognition and scene analysis.* RO Duda, PE Hart - 1973.

3 Code

```
% main
clear;
%% input
% image
filename = 'cheetah.bmp';
img = imread(filename);
imshow(img)
img = double(img)/255;
filename = 'cheetah_mask.bmp';
img_gt = imread(filename);
img_gt = im2double(img_gt);
% pattern
ZZ_ptn = load('Zig-Zag Pattern.txt');
ZZ_ptn = ZZ_ptn(:) + 1;
% Training data
trainSample = load('TrainingSamplesDCT_subsets_8.mat');
TrSamp_BG_set = {trainSample.D1_BG, trainSample.D2_BG, ...
    trainSample.D3_BG, trainSample.D4_BG};
TrSamp_FG_set = {trainSample.D1_FG, trainSample.D2_FG, ...
    trainSample.D3_FG, trainSample.D4_FG};

%% crop into 8x8 blocks
fun = @(blk) getDctFromBlk(blk, ZZ_ptn);
img_blk = nlfilter(img,[8 8],fun);
height = size(img,1);
width = size(img,2);

img_blk = img_blk(:);
img_blk = cell2mat(img_blk);

% imshow(uint8(img_blk))



%% Bayes-BDR Estimate P_x|D
% Strategy 1: P_mu
prior = load('prior_1.mat'); prior_name = 'pri_1';
% prior = load('prior_2.mat'); prior_name = 'pri_2';
alpha = load('alpha.mat');
alpha_all = alpha.alpha;
FG.mu0 = prior.mu0_FG;
BG.mu0 = prior.mu0_BG;
w0 =  diag (prior.W0);

% loop through Dataset
for j = 1:size(TrSamp_BG_set, 2)
%% class prior probability
TrSamp_BG = cell2mat(TrSamp_BG_set(j));
TrSamp_FG = cell2mat(TrSamp_FG_set(j));
num_TrSamp = size(TrSamp_BG, 1) + size(TrSamp_FG, 1);
p_y_FG = size(TrSamp_FG, 1)/num_TrSamp;
p_y_BG = size(TrSamp_BG, 1)/num_TrSamp;
% prior 'Y'
```

```
Prior_cls.p_y_BG = p_y_BG;
Prior_cls.p_y_FG = p_y_FG;
errEst = zeros(1, size(alpha_all, 2));
% compute sample mu and var
% BG
features = TrSamp_BG;
BG = MLEstimation(BG, features);
% FG
features = TrSamp_FG;
FG = MLEstimation(FG, features);

%% Bayes estimation
FG.cov  = FG.covSpl;
BG.cov  = BG.covSpl;
% loop through different alpha
for i = 1:size(alpha_all, 2)
    alpha = alpha_all(i);
    cov0 = alpha*w0;
    FG.cov0 = cov0;
    BG.cov0 = cov0;
    % compute probability
    features_est = img_blk;
    % FG
    FG = bayesEstimation(FG);
    Par = FG;
    Par.p_x_D = mvnpdf(features_est, Par.mu_n, Par.cov_all);
    FG = Par;
    % BG
    BG = bayesEstimation(BG);
    Par = BG;
    Par.p_x_D = mvnpdf(features_est, Par.mu_n, Par.cov_all);
    BG = Par;
    % seg
    img_seg = FG.p_x_D*p_y_FG > BG.p_x_D*p_y_BG;
    img_seg = reshape(img_seg, height, width);
    filename_result = strcat('result_bayes', '_a_', int2str(i),'.png');
    imagesc(img_seg)
    colormap(gray(255));
    imwrite(img_seg, filename_result)
    % err
    errEst(i) = estimationError(img_gt, img_seg, Prior_cls);
end

errors.bayes = errEst;

%% Bayes-MAP estimate
for i = 1:size(alpha_all, 2)
    alpha = alpha_all(i);
    cov0 = alpha*w0;
    FG.cov0 = cov0;
    BG.cov0 = cov0;
    % compute probability
    features_est = img_blk;
    % FG
```

```
    FG = bayesEstimation(FG);
    % BG
    BG = bayesEstimation(BG);
    % seg
    features = img_blk;
    p_x_y_FG = mvnpdf(features, FG.mu_n, FG.cov_n);
    p_x_y_BG = mvnpdf(features, BG.mu_n, BG.cov_n);
    %%%% test
    p_x_y_FG = mvnpdf(features, FG.mu_n, FG.covSpl);
    p_x_y_BG = mvnpdf(features, BG.mu_n, BG.covSpl);

    img_seg = p_x_y_FG*p_y_FG > p_x_y_BG*p_y_BG;
    img_seg = reshape(img_seg, height, width);
    % save
    filename_result = strcat('result_bayes', '_a_', int2str(i),'_MAP.png');
    imagesc(img_seg)
    colormap(gray(255));
    imwrite(img_seg, filename_result)
    % err
    errEst(i) = estimationError(img_gt, img_seg, Prior_cls);
end
errors.MAP = errEst;

%% ML-BDR
filename_result = 'result_64f.png';
% compute probability
features = img_blk;
p_x_y_FG = mvnpdf(features, FG.muSpl, FG.covSpl);
p_x_y_BG = mvnpdf(features, BG.muSpl, BG.covSpl);
img_seg = p_x_y_FG*p_y_FG > p_x_y_BG*p_y_BG;
img_seg = reshape(img_seg, height, width);
err = estimationError(img_gt, img_seg, Prior_cls);
errors.ML = errEst.*0 + err;

% plot
figure
hold on
plot(alpha_all, errors.bayes, 'r');
plot(alpha_all, errors.MAP, 'g');
plot(alpha_all, errors.ML, 'b');
set(gca, 'XScale', 'log');
xlabel('Alpha (log scale)');
ylabel('Probability of Error');
legend('Bayes Estimate', 'MAP Estimate', 'ML Estimate')
% save plot
plot_name = strcat('err_D', int2str(j),'_', prior_name, '.png');
saveas(gcf, plot_name);
hold off
end




%% functions
```

```matlab

function error = estimationError (img_gt, img_seg, Priors)
mask = img_gt;
error_FG = abs(img_gt - img_seg).*mask;
error_FG = sum(error_FG(:))/sum(mask(:));
mask = abs(img_gt-1);
error_BG = abs(img_gt - img_seg).*mask;
error_BG = sum(error_BG(:))/sum(mask(:));
error = error_BG*Priors.p_y_BG + error_FG*Priors.p_y_FG;
end

function Par = MLEstimation (Par, features)
    Par.muSpl = mean(features, 1);
    Par.n = size(features, 1);
    Par.covSpl = cov(features);
end

function Par = bayesEstimation (Par)
    Par.mu_n_hat = Par.muSpl;
    Par.invCov0Cov = inv(Par.cov0 + 1/Par.n*Par.cov);
    Par.mu_n = (Par.cov0*Par.invCov0Cov*Par.mu_n_hat' + ...
        1/Par.n*Par.cov*Par.invCov0Cov*Par.mu0')';
    Par.cov_n = Par.cov0*Par.invCov0Cov*1/Par.n*Par.cov;
    Par.cov_all = Par.cov + Par.cov_n;
%     Par.p_x_D = mvnpdf(features, Par.mu_n, Par.cov + Par.cov_n);
end

function plot2FeaturePdf(phat1, color1, lb1, phat2, color2, lb2, ...
    plot_name, features)
num_plots = min(size(phat1, 1), size(phat2,1));
x_num = 4;
y_num = num_plots/x_num;
% y_num = 4;
for i = 1:num_plots
    subp = subplot(y_num, x_num, mod(i-1,x_num*y_num)+1, 'replace');
    offset = (i-1)*2;
    % plot 1
    p = phat1(i, :);
    mu = p(1);
    s = p(2);
    x = (mu - 3*s):0.001:(mu + 3*s);
    y = normpdf(x,mu,s);
    p1 = plot(x,y,color1,'LineWidth',1);
    hold on
    % plot 2
    p = phat2(i, :);
    mu = p(1);
    s = p(2);
    x = (mu - 3*s):0.001:(mu + 3*s);
    y = normpdf(x,mu,s);
    p2 = plot(x,y,color2,'LineWidth',1);
    title(subp, int2str(features(i)))
    % save
    if mod(i, x_num*y_num)+1 == 1
```

```
        legend([p1,p2], [lb1, lb2]);
        savefig(strcat(plot_name, int2str(i),'.fig'));
        hold off
    end
end
end

function phat_all = mleFeatures(features, plot_name)
phat_all = [];
for i = 1:size(features,2)
    feature = features(:,i);
    [phat,pci] = mle(feature);
    phat_all = cat(1, phat_all, phat);
    x_num = 4;
    y_num = 4;
    subp = subplot(y_num, x_num, mod(i-1,x_num*y_num)+1, 'replace');
    histogram(feature,'Normalization','pdf')
    hold on
    x = min(feature):0.001:max(feature);
    y = normpdf(x,phat(1),phat(2));
    plot(x,y,'r','LineWidth',2)
    title(subp, i)
    if mod(i, x_num*y_num)+1 == 1
        savefig(strcat(plot_name, int2str(i),'.fig'));
        hold off
    end
end
end

function isFG = cmpProbFGBGMulG(feature, mulg_FG, mulg_BG, p_y_FG, p_y_BG)
    size(feature)
    p_x_y_FG = mvnpdf(feature, mulg_FG.mu, mulg_FG.cov);
    p_x_y_BG = mvnpdf(feature, mulg_BG.mu, mulg_BG.cov);
    if(p_x_y_FG*p_y_FG > p_x_y_BG*p_y_BG)
        isFG = 1;
    elseif (p_x_y_FG*p_y_FG == p_x_y_BG*p_y_BG)
        isFG = 0;
%          "alert!"
    else
        isFG = 0;
    end
end

% input: matrix of features, and ZZ pattern
% output: features map
function features = getFeaturesFromMat(mat, ZZ_ptn)
    features = zeros(size(mat, 1), 1);
    for i = 1:size(mat,1)
        idx = get2MaxIdx(mat(i,:));
%          features(i) = ZZIdxFromIdx(idx, ZZ_ptn);
        features(i) = idx;
    end
end

```

```
% get Dct vector from blocks
function f = getDctFromBlk(blk, ZZ_ptn)
f = dct2(blk);
[¬,ZZsort]=sort(ZZ_ptn);
f = f(:);
f =f(ZZsort);
f = {reshape(f, [1 size(f)])};
end

% get 2nd largest value and index from blocks
function idx = get2MaxDctFromBlk(blk, ZZ_ptn)
f = dct2(blk);
idx = get2MaxIdx(f(:));
idx = ZZ_ptn(idx);
end


function ZZIdx = ZZIdxFromIdx(idx, ZZ_ptn)
ZZIdx = ZZ_ptn(idx);
end

% get the index of 2nd largest number in an array
function idx = get2MaxIdx(arr)
f = abs(arr(:));
[v, idx] = max(f);
f(idx) = 0;
[v, idx] = max(f);
end
```