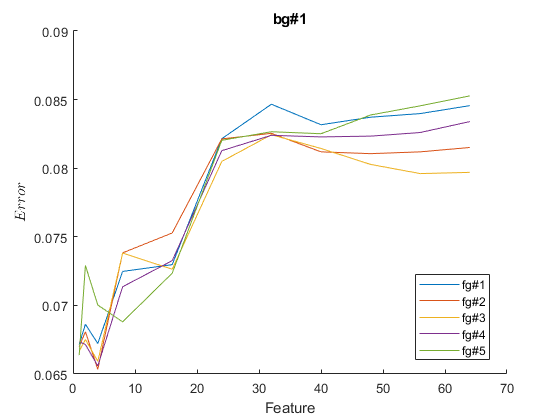
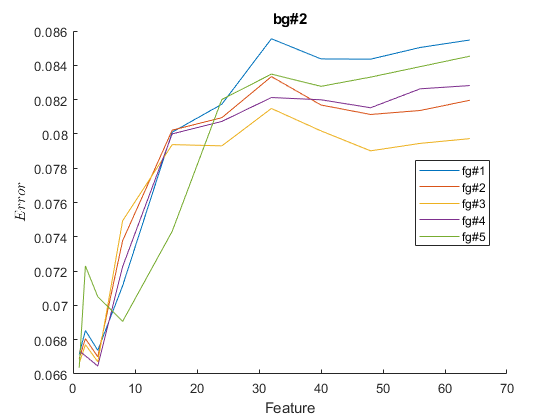
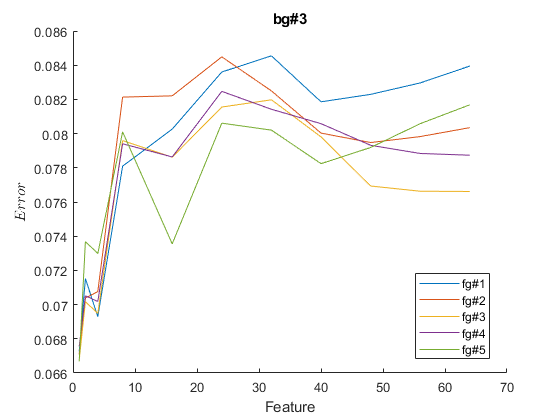
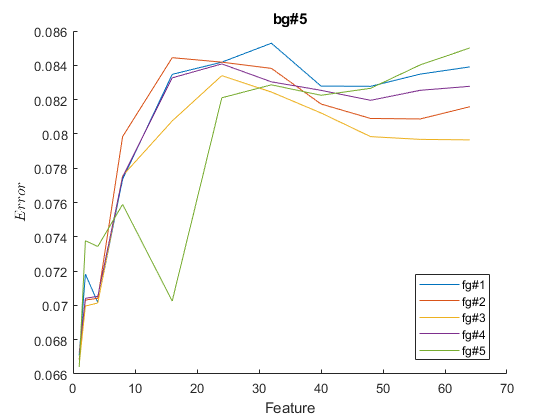
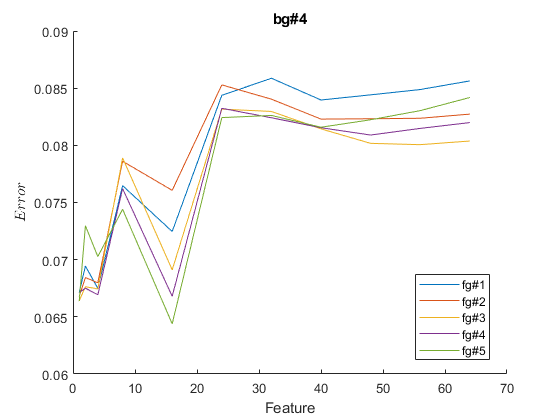
Kalvin Goode

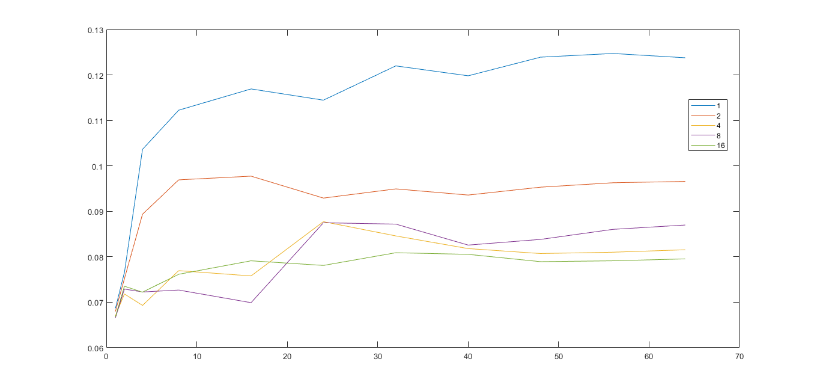
“bg” means background, grass data

“fg” means foreground, cheetah data

Above are the 5 plots containing dimension versus the probability of errors from the combination of 5 Gaussian mixtures and 8 components for each class. In all plots, we see that the error increases from 6.5% to 8.6% and contains a lot of variation in between. All plots show that the error is the lowest when only take 1 dimension for classification. I believe it is either because the poor initialization or improper EM algorithm was used for higher dimension. Notice that the error rate of bg#4 has a sudden drop when dimension is 16, which is different than other plots. It can probably consider as outliers. Overall, I believe we can reduce additional error for high dimension by increase iteration or examining other possible initialization for mean and variance. Nevertheless, EM seems to have better results for classifying this cheetah image compare to other techniques we learned in class.



Above is the plot for the dimension vs error rate with Gaussian mixtures that have 1. 2, 4, 8 or 16 components. Similar to the previous task, the error rate increases a as dimension increases. However, we see that as components increase, the result of error rates seems to be more consistent and less variation throughout different dimension. On the other hand, Gaussian mixture model with 1 component has the worst result in classifying the image. This is an example of bias-variance tradeoff in predictive model. Overall, testing with various components, the results had shown that mixture model technique does have its significance among other machine learning techniques, and it is worth to apply it in the future.

Appendix

load('TrainingSamplesDCT\_8.mat');

zig=load('Zig-Zag Pattern.txt')+1;

truth=imread('cheetah\_mask.bmp');

truth=im2double (truth);

pri\_f=size(TrainsampleDCT\_FG,1)...

/(size(TrainsampleDCT\_BG,1)+size(TrainsampleDCT\_FG,1));

pri\_b=1-pri\_f;

cheetah= im2double(imread('cheetah.bmp'));

cheetah\_p=padarray(cheetah,[4,3],0,'pre');

cheetah\_p=padarray(cheetah\_p,[3,4],0,'post');

n=1;

cheetah\_dct=zeros(68850,64);

for i=1:size(cheetah\_p,1)-7

for j=1:size(cheetah\_p,2)-7

temp=dct2(cheetah\_p(i:i+7, j:j+7));

for k=1:8

for m=1:8

cheetah\_dct(n,zig(k,m))=temp(k,m);

end

end

n=n+1;

end

end

dims=[1 2 4 8 16 24 32 40 48 56 64];

n\_comp=8; %components

n\_mix=5;

mean\_bg=zeros(n\_mix, n\_comp,64);

mean\_fg=zeros(n\_mix, n\_comp,64);

cov\_bg=zeros(n\_mix, n\_comp,64,64);

cov\_fg=zeros(n\_mix, n\_comp,64,64);

pi\_bg=zeros(n\_mix, n\_comp); %weight, pi\_c

pi\_fg=zeros(n\_mix, n\_comp);

for i=1:n\_mix

[mean\_bg(i,:,:),cov\_bg(i,:,:,:),pi\_bg(i,:)]=...

take\_em(TrainsampleDCT\_BG,n\_comp,64,250);

[mean\_fg(i,:,:),cov\_fg(i,:,:,:),pi\_fg(i,:)]=...

take\_em(TrainsampleDCT\_FG,n\_comp,64,250);

end

im\_set=zeros(n\_mix,n\_mix,size(dims,2),...

size(cheetah,1),size(cheetah,2));

err\_set=zeros(n\_mix,n\_mix,size(dims,2),1);

for b=1:n\_mix

for f=1:n\_mix

for d=1:size(dims,2)

[im\_set(b,f,d,:,:), err\_set(b,f,d)]=...

take\_im(cheetah\_dct,...

cheetah,pri\_b,pri\_f,truth,dims(d),...

mean\_bg(b,:,:),cov\_bg(b,:,:,:),pi\_bg(b,:),...

mean\_fg(f,:,:),cov\_fg(f,:,:,:),...

pi\_fg(f,:),n\_comp,1);

end

end

end

figure();

for i=1:n\_mix

for j=1:n\_mix

err=zeros(size(dims,2),1);

for k=1:size(dims,2)

err(k)=err\_set(i,j,k);

end

hold on;

plot(dims,err)

end

xlabel('Feature');

ylabel({'$Error$'},'Interpreter','latex');

title(['bg#' int2str(i)])

legend('fg#1','fg#2','fg#3','fg#4','fg#5','location','best')

hold off;

figure();

end

n\_comp=[1 2 4 8 16];

mean\_bg2=cell(1,size(n\_comp,2));

cov\_bg2=cell(1,size(n\_comp,2));

pi\_bg2=cell(1,size(n\_comp,2));

mean\_fg2=cell(1,size(n\_comp,2));

cov\_fg2=cell(1,size(n\_comp,2));

pi\_fg2=cell(1,size(n\_comp,2));

for i=1:size(n\_comp,2)

[mean\_bg2{i},cov\_bg2{i},pi\_bg2{i}]=...

take\_em(TrainsampleDCT\_BG,n\_comp(i),64,50);

[mean\_fg2{i},cov\_fg2{i},pi\_fg2{i}]=...

take\_em(TrainsampleDCT\_FG,n\_comp(i),64,50);

end

im\_set2=cell(size(dims,2));

for i=1:size(n\_comp,2)

for d=1:size(dims,2)

[im\_set2{d}, err\_set2(i,d)]=...

take\_im(cheetah\_dct,...

cheetah,pri\_b,pri\_f,truth,dims(d),...

mean\_bg2{i},cov\_bg2{i},pi\_bg2{i},...

mean\_fg2{i},cov\_fg2{i},...

pi\_fg2{i},n\_comp(i),2);

end

end

for i =1:size(n\_comp,2)

plot(dims,err\_set2(i,:))

hold on;

end

legend('1','2','4','8','16','location','best')

disp("end")

function [mu,sig,pi]=take\_em(dct,n\_c,dim,iter)

%initialization

mu=-2.5+5\*rand(n\_c,dim);

cov\_t=cell(1,n\_c);

for k=1:n\_c

cov\_t{k}=diag(1+3\*rand(dim,1));

end

pi=ones(1,n\_c);

pi=pi/sum(pi);

%EM

for i=1:iter

%E

for k=1:size(dct,1)

t\_x=zeros(1,n\_c);

for j=1:n\_c

t\_x(j)=mvnpdf(dct(k,:),mu(j,:),cov\_t{j})\*pi(j);

end

t(k,:)=t\_x/sum(t\_x);

end

pi=sum(t)/size(dct,1);

%Maximize

for k = 1: n\_c

mu(k,:) = sum(dct.\* t(:, k))/sum(t(:, k));

cov\_t{k}=diag(diag((dct - mu(k,:)).\* (t(:, k))' \* (dct-mu(k,:)) /sum(t(:, k)))+0.00001);

end

end

sig=zeros(k,64,64);

for k=1:n\_c

sig(k,:,:)=cov\_t{k};

end

end

function [image, err]=take\_im(cheetah\_dct,...

cheetah,pri\_b,pri\_f,truth,dim,...

mean\_bg,cov\_bg,pi\_bg,mean\_fg,cov\_fg,pi\_fg,n\_c,p)

like\_b=0;

like\_f=0;

for i=1:n\_c %components

c\_bg=zeros(dim,dim);

c\_fg=zeros(dim,dim);

mn\_bg=zeros(1,dim);

mn\_fg=zeros(1,dim);

if(p==1)

for j=1:dim

for k=1:dim

c\_bg(j,k)=cov\_bg(1,i,j,k);

c\_fg(j,k)=cov\_fg(1,i,j,k);

end

end

for j=1:dim

mn\_bg(1,j)=mean\_bg(1,i,j);

mn\_fg(1,j)=mean\_fg(1,i,j);

end

else

for j=1:dim

for k=1:dim

c\_bg(j,k)=cov\_bg(i,j,k);

c\_fg(j,k)=cov\_fg(i,j,k);

end

end

for j=1:dim

mn\_bg(1,j)=mean\_bg(i,j);

mn\_fg(1,j)=mean\_fg(i,j);

end

end

like\_b=like\_b+pi\_bg(i)\*mvnpdf(cheetah\_dct(:,1:dim),...

mn\_bg,c\_bg);

like\_f=like\_f+pi\_fg(i)\*mvnpdf(cheetah\_dct(:,1:dim),...

mn\_fg,c\_fg);

end

image=zeros(size(cheetah,1),size(cheetah,2));

%disp(like\_b(1:10))

n=1;

for i=1:size(cheetah,1)

for j=1:size(cheetah,2)

if(like\_b(n)\*pri\_b>=like\_f(n)\*pri\_f)

image(i,j)=0;

else

image(i,j)=1;

end

n=n+1;

end

end

%calculate error

err=error(image,truth);

end

function err=error(img,truth)

%calculate error

err=0;

for i=1:size(truth,1)

for j=1:size(truth,2)

if (img(i,j)~= truth(i,j))

err=err+1;

end

end

end

err=err/(size(truth,1)\*size(truth,2));

end