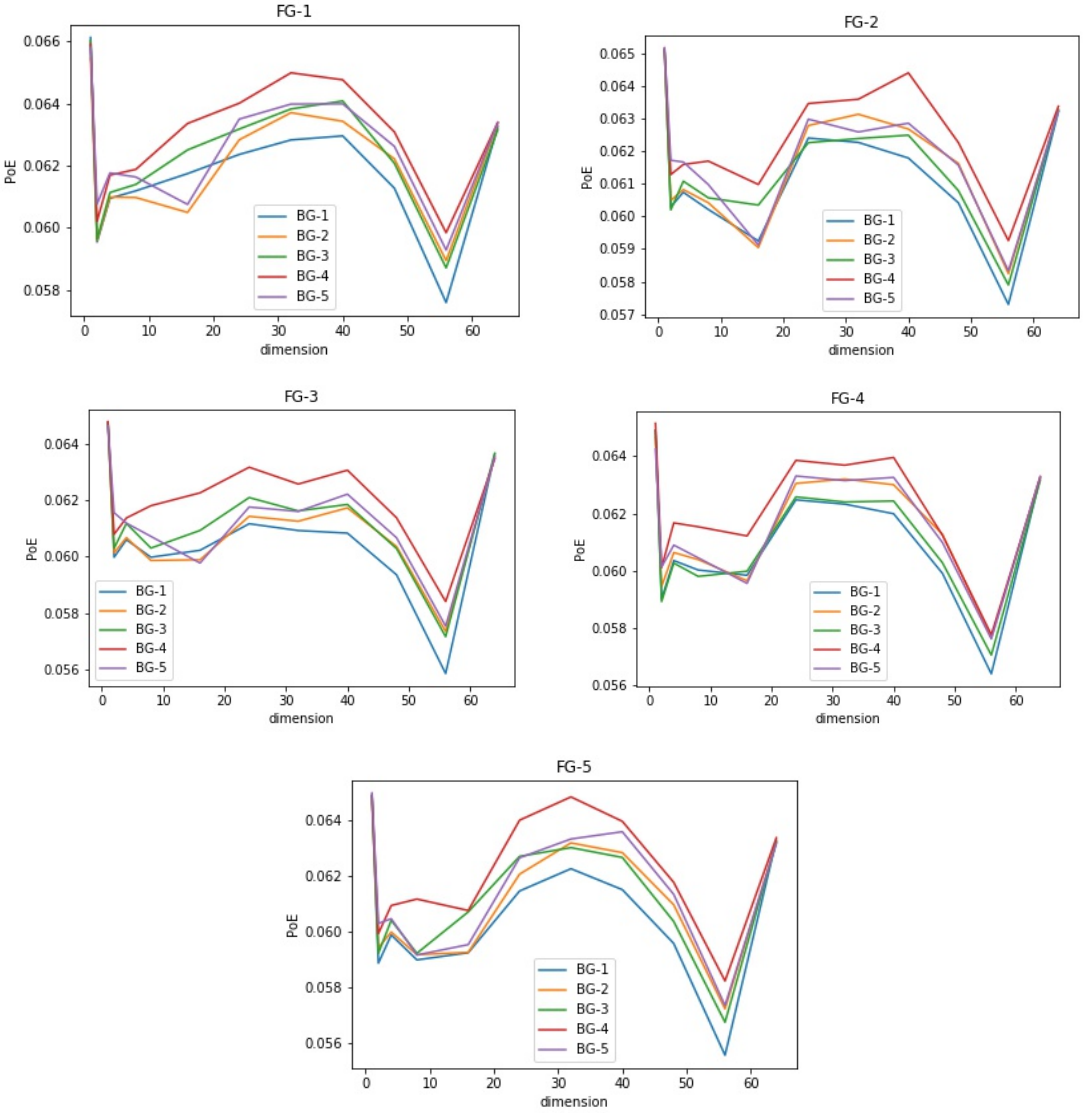
# Homework 5


## Yan Sun

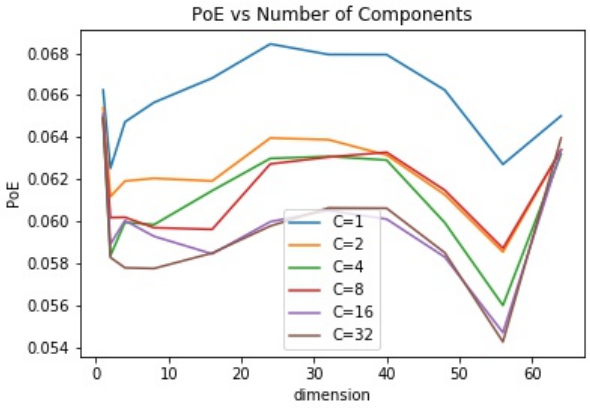
## A53240727


## Problem 1



For previous homework it has been concluded that using selected best 8 features works better than using all 64 features. In this way, using all 64 features will not be the best solution. In addition, only using one feature is obviously not the best solution since it lose too much information. In this way, the lowest PoE will appear between 1-64 dimension when we select the comparably better combination of features. The initialization could has influence since the initialization could make the mixture converges to different distribution, which could also make difference for the later prediction.

## Problem 2




It could be observed that increasing the number of components will assist in decreasing the probability of error for most of the situation with different number of feature dimension. With more components there will be higher probability to get a better fitting for the ground truth.

Code:

```
load('TrainingSamplesDCT_8_new.mat');
train_FG = TrainsampleDCT_FG;
train_BG = TrainsampleDCT_BG;

num_comp_list = [1,2,4,8,16,32];

for idx_comp=1:size(num_comp_list,2)
   % Configuration
   num_mixtures = 1;
   MAX_LOOP = 200;
   num_comp = num_comp_list(idx_comp);
   dim_list = [1 2 4 8 16 24 32 40 48 56 64];

   mixture_cov_FG = containers.Map;
   mixture_mu_FG = containers.Map;
   mixture_pi_FG = containers.Map;
   mixture_cov_BG = containers.Map;
   mixture_mu_BG = containers.Map;
   mixture_pi_BG = containers.Map;

   % Train Gaussian Mixtures for FG class

   disp(['Train Mixtures for FG starts !'])

   for idx_mix=1:num_mixtures
      % Initialize
      cov_c = containers.Map;
      mu_c = [];
      pi_c = [];
      for c=1:num_comp
         tmp1 = normrnd(5,0.3,[1 64])+1;
         tmp2 = normrnd(5,0.3,[1 64]);
         cov_c(int2str(c)) = diag(tmp1 ./ sum(tmp1));
         mu_c = [mu_c; transpose(tmp2 ./ sum(tmp2))];
         pi_c = [pi_c 1/num_comp];
      end

      % EM Train Loop
      for step=1:MAX_LOOP
         hij = [];
         for j=1:num_comp
            hij = [hij mvnpdf(train_FG,mu_c(j,:),cov_c(int2str(j))).*pi_c(j)];
         end
         hij = hij ./ sum(hij,2);
         hij_sum = sum(hij,1);

         % Update
         % Update pi_c
         pi_c = hij_sum ./ size(train_FG,1);

         % Update cov_c
         for j=1:num_comp
            tmp_comp = sum((train_FG-mu_c(j,:)).^2 .* hij(:,j),1) ./ sum(hij(:,j));
            cov_c(int2str(j)) = diag(tmp_comp);
         end

         % Update mu_c
         mu_c = [];
         for j=1:num_comp
            tmp_comp = sum(hij(:,j).*train_FG,1) ./ sum(hij(:,j));
            mu_c = [mu_c; tmp_comp];
         end
      end

      % Save Train Result
      mixture_cov_FG(int2str(idx_mix)) = cov_c;
      mixture_mu_FG(int2str(idx_mix)) = mu_c;
      mixture_pi_FG(int2str(idx_mix)) = pi_c;
```

```
end

disp(['Train Mixtures for FG finished !'])

% Train Gaussian Mixtures for BG class

disp(['Train Mixtures for BG starts !'])

for idx_mix=1:num_mixtures
   % Initialize
   cov_c = containers.Map;
   mu_c = [];
   pi_c = [];
   for c=1:num_comp
      tmp1 = normrnd(5,0.1,[1 64])+5;
      tmp2 = normrnd(5,0.1,[1 64]);
      cov_c(int2str(c)) = diag(tmp1 ./ sum(tmp1));
      mu_c = [mu_c; transpose(tmp2 ./ sum(tmp2))];
      pi_c = [pi_c 1/num_comp];
   end

   % EM Train Loop
   for step=1:MAX_LOOP
     hij = [];
     for j=1:num_comp
       hij = [hij mvnpdf(train_BG,mu_c(j,:),cov_c(int2str(j))).*pi_c(j)];
     end
     hij = hij ./ sum(hij,2);
     hij_sum = sum(hij,1);

     % Update
     % Update pi_c
     pi_c = hij_sum ./ size(train_BG,1);

     % Update cov_c
     for j=1:num_comp
       tmp_comp = sum((train_BG-mu_c(j,:)).^2 .* hij(:,j),1) ./ sum(hij(:,j));
       cov_c(int2str(j)) = diag(tmp_comp);
     end

     % Update mu_c
     mu_c = [];
     for j=1:num_comp
       tmp_comp = sum(hij(:,j).*train_BG,1) ./ sum(hij(:,j));
       mu_c = [mu_c; tmp_comp];
     end
   end

   % Save Train Result
   mixture_cov_BG(int2str(idx_mix)) = cov_c;
   mixture_mu_BG(int2str(idx_mix)) = mu_c;
   mixture_pi_BG(int2str(idx_mix)) = pi_c;
end

disp(['Train Mixtures for BG Finished !'])

% Predict by BDR
img = imread('cheetah.bmp');
img = im2double(img);

% Add paddle
img = [img zeros(size(img,1),7)];
img = [img; zeros(7, size(img,2))];
[m,n] = size(img);

prior_FG = 250/(250+1053);
prior_BG = 1053/(250+1053);
error_container = containers.Map;

for idx_fg=1:num_mixtures
```

```
cov_FG = mixture_cov_FG(int2str(idx_fg));
mu_FG = mixture_mu_FG(int2str(idx_fg));
pi_FG = mixture_pi_FG(int2str(idx_fg));
error_list = [];
for idx_bg=1:num_mixtures
    tmp_error_list = [];
    cov_BG = mixture_cov_BG(int2str(idx_bg));
    mu_BG = mixture_mu_BG(int2str(idx_bg));
    pi_BG = mixture_pi_BG(int2str(idx_bg));
    total_blocks = containers.Map;

    for idx_dim=1:size(dim_list,2)
      total_blocks(int2str(idx_dim)) = zeros(m-7,n-7);
    end

    for idx_dim = 1:size(dim_list,2)
       tmp_block = total_blocks(int2str(idx_dim));
       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(1:idx_dim);
             g_cheetah = zeros(1,num_comp);
             g_grass = zeros(1,num_comp);
             for idx_comp=1:num_comp
                ave_tmp_FG = mu_FG(idx_comp,:);
                ave_tmp_BG = mu_BG(idx_comp,:);

                % cheetah
                sigma_cheetah = cov_FG(int2str(idx_comp));
                tmp_cheetah = mvnpdf(feature,ave_tmp_FG(1:idx_dim),sigma_cheetah(1:idx_dim,1:idx_dim));

                % grass
                sigma_grass = cov_BG(int2str(idx_comp));
                tmp_grass = mvnpdf(feature,ave_tmp_BG(1:idx_dim),sigma_grass(1:idx_dim,1:idx_dim));

                g_cheetah(idx_comp) = pi_FG(idx_comp) * tmp_cheetah;
                g_grass(idx_comp) = pi_BG(idx_comp) * tmp_grass;
             end
             if sum(g_cheetah) * prior_FG >= sum(g_grass) * prior_BG
                tmp_block(i,j) = 1;
             end
          end
       end
       total_blocks(int2str(idx_dim)) = tmp_block;
       imwrite(total_blocks(int2str(idx_dim)), ['Prediction_' 'num_comp_' int2str(num_comp) '_dim_' int2str(idx_dim) '.jpg']);
    end

    ground_truth = imread('cheetah_mask.bmp')/255;
    x = size(ground_truth, 1);
    y = size(ground_truth, 2);

    %%% save prediction
    for idx_dim=1:size(dim_list,2)
       prediction = mat2gray(total_blocks(int2str(idx_dim)));
       count1 = 0;
       count2 = 0;
       count_cheetah_truth = 0;
       count_grass_truth = 0;
       for i=1:x-7
          for j=1:y-7
             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 = (count1/count_cheetah_truth) * prior_FG;
        error2 = (count2/count_grass_truth) * prior_BG;
        total_error = error1 + error2;
        disp(['Error = ' num2str(total_error)]);
        tmp_error_list = [tmp_error_list total_error];
    end
    disp(['-------']);
    error_list = [error_list; tmp_error_list];
  end
  error_container(int2str(idx_fg)) = error_list;
 end
end
```