ECE271A HW3 Report

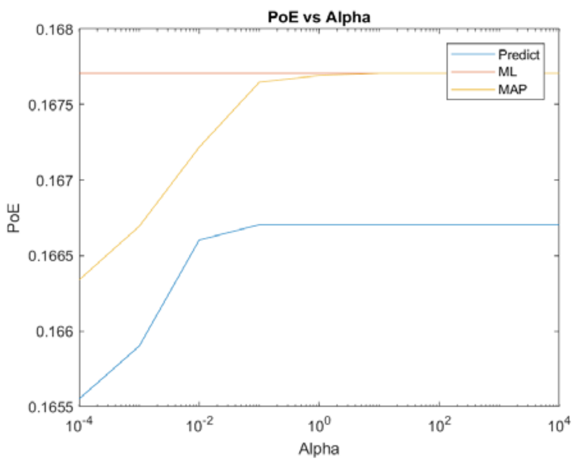
Yan Sun

A53240727

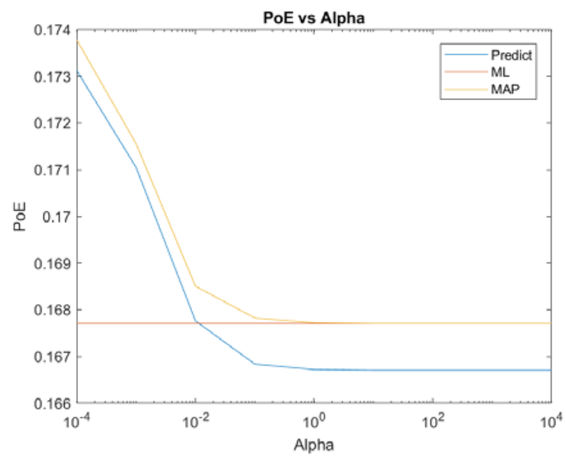
Computer Problem

(The results are shown in the following pictures. I will show the plots firstly and then answer each question.)

Dataset1

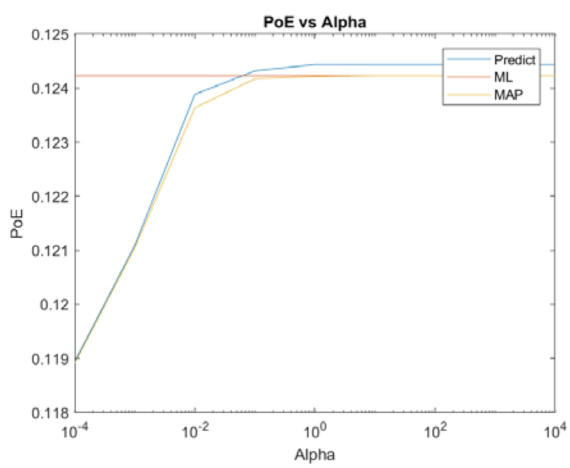


Strategy1

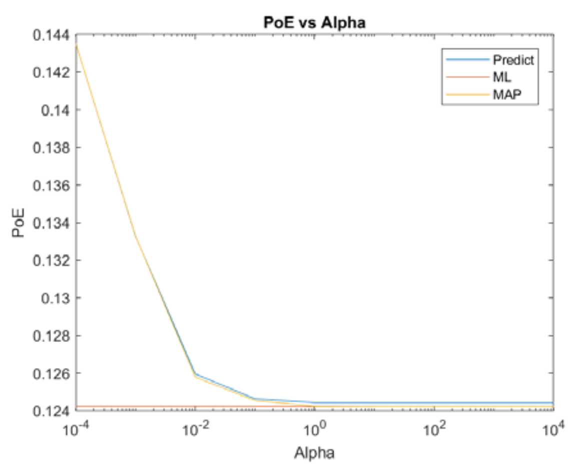


Strategy2

Dataset2

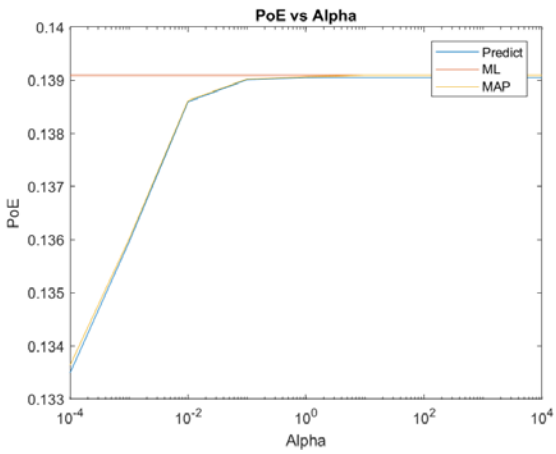


Strategy1

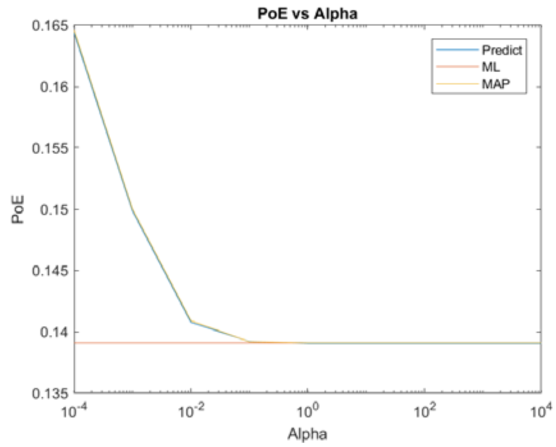


Strategy2

Dataset3

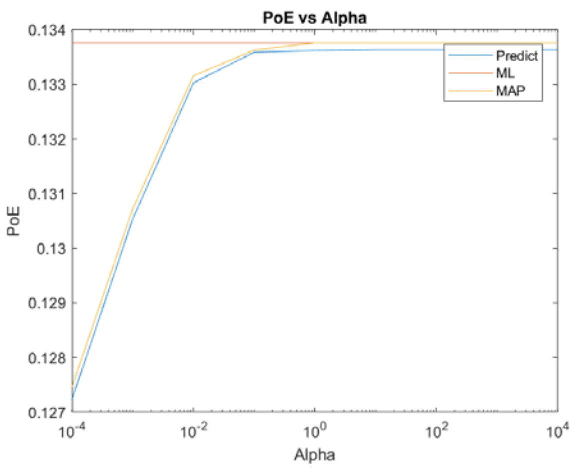


Strategy1

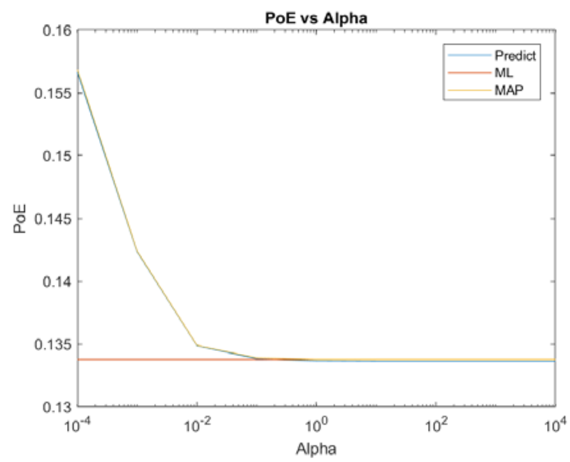


Strategy2

Dataset4



Strategy1



Strategy2

(1) The relative behavior of these three curves:

Observation:

In this problem, I use dataset one and strategy one to plot the curve of probability of error (PoE) versus alpha. The PoE of dataset one for both strategies is around 16%~17%. In this curve, PoE increases as alpha increases obviously when alpha is small. When alpha is large, however, the PoE tend to be fixed at a constant value.

$$(\mathbf{\Sigma}_0)_{ii} = \alpha w_i.$$

$$\boldsymbol{\mu}_n = \boldsymbol{\Sigma}_0\left(\boldsymbol{\Sigma}_0 + \frac{1}{n}\boldsymbol{\Sigma}\right)^{-1}\hat{\boldsymbol{\mu}}_n + \frac{1}{n}\boldsymbol{\Sigma}\left(\boldsymbol{\Sigma}_0 + \frac{1}{n}\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\mu}_0$$

$$\boldsymbol{\Sigma}_n = \boldsymbol{\Sigma}_0\left(\boldsymbol{\Sigma}_0 + \frac{1}{n}\boldsymbol{\Sigma}\right)^{-1}\frac{1}{n}\boldsymbol{\Sigma}.$$

Explain:

When alpha is small, based on the intuitive combination of mean value, it could be found that the prior information has more weights within the total μ value considered. At this time, MAP and Prediction model share the same mu value and they tend to have similar PoE value. The Σn term will be small so that the contribution of prior covariance is small in Σ+Σn, which also sontributed to the phenomenon that MAP and Bayes predictive model's PoE tends to be close. The difference of PoE between ML and Bayes is large, which is mainly caused by the large difference of μ used in different methods.

When alpha becomes large enough, in the equation of μn, the term with primitive μ (μ hat) will have more weights. At this time, the PoE of MAP tends to be close to ML since they already share the same covariance. In the end, when alpha is very large, the ML part in Bayes predictive model has enough weights so that the PoE of Bayes is comparably close to ML compared to the time when alpha is small. Although the PoE seems fixed at this time, the prior information is also considered to be part of Bayes predictive model so that their performance will be different based on the dataset who provided the prior information.

(2) How that behavior changes from dataset to dataset:

Observation:

It could be observed that in dataset one, Bayes predictive model is obviously performed better than ML and MAP. In dataset two, Bayes predictive model is obviously performed worse than ML and MAP. In dataset three and four, Bayes predictive model just performs slightly better than ML and MAP.

Explain:

The difference of dataset has influence on prior information, which is showed in the equation when we try to get the prior μ or Σ. To be specific, the size of dataset is the primary reason since the value of N in those equations will make a difference. Dataset three and four has more data than dataset one and dataset two so the prior information provided by them will not count much (in the expression of μn and Σn, the 1/n term will make the prior information less importance so that the big size of data will make the ML part more weighted instead). In this way, when alpha is large enough, Bayes predictive models' performance will be closer to ML and Map compared to dataset one and dataset two. Dataset one and dataset two has comparably smaller size of data than dataset three and dataset four. In this way, their Bayes predictive model performance will tend to depend on the prior information provided by specific dataset so that the ultimate performance when alpha is large enough will be either better or worse than ML and MAP obviously.

(3) How all of the above change when strategy 1 is replaced by strategy 2:

Observation:

In both strategies, the PoE of ML is fixed. In strategy one, the PoE of MAP and Bayes increases as alpha increases while in strategy two the PoE of MAP and Bayes decreases as alpha increases. When alpha is large enough, all the PoE values tend to be fixed and the relative performance stays the same for same dataset on these two strategies.

Explain:

The different trend is caused by the different definition of prior information defined in two strategies. As aforementioned explanation mentioned in (1), when alpha is small, prior information has more weights for the Classifier. However, strategy one gives two classes different mean while strategy two gives two classes same mean. It is obviously that these two classes should be weighted differently so that strategy two's prior information is worse that strategy one. In this way, when alpha is small, strategy one's prior information makes Bayes and MAP better than ML while strategy two's prior information makes Bayes and MAP worse than ML. When alpha is large enough, all the models' performance under strategy two tend to be close to what it is under strategy one since the ML part in Bayes and MAP ted to be more weighted. The relative performance among three methods are also similar to what it is under strategy one.

Code:

The total running time could be around 15 minutes. It may be varied depends on the device and MATLAB version.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
train_set = load('TrainingSamplesDCT_subsets_8.mat');
alpha = load('Alpha.mat');
alpha = alpha.alpha;

for strategy_idx = 1:2
   if strategy_idx == 1
      strategy = load('Prior_1.mat');
   elseif strategy_idx == 2
      strategy = load('Prior_2.mat');
   end

   for dataset_idx = 1:4
      if dataset_idx == 1
         d1_BG = train_set.D1_BG;
         d1_FG = train_set.D1_FG;
      elseif dataset_idx == 2
         d1_BG = train_set.D2_BG;
         d1_FG = train_set.D2_FG;
      elseif dataset_idx == 3
         d1_BG = train_set.D3_BG;
         d1_FG = train_set.D3_FG;
      elseif dataset_idx == 4
         d1_BG = train_set.D4_BG;
         d1_FG = train_set.D4_FG;
      end

      bayes_error = [];
      mle_error = [];
      map_error = [];
      n_FG = size(d1_FG,1);
      n_BG = size(d1_BG,1);


      % Loop for different alpha
      for alpha_idx = 1:size(alpha,2)
```

```
    cov_0 = zeros(64,64);
    for idx = 1:64
        cov_0(idx,idx) = alpha(alpha_idx)*strategy.W0(idx);
    end

    % FG
    d1_FG_cov = cov(d1_FG) * (n_FG-1)/ n_FG;
    tmp2 = inv(cov_0 + (1/ n_FG)*d1_FG_cov);
    mu_1_FG = cov_0 * tmp2 * transpose(mean(d1_FG)) + (1/ n_FG) * d1_FG_cov * tmp2 *
transpose(strategy.mu0_FG);
    cov_1_FG = cov_0 * tmp2 * (1/ n_FG) * d1_FG_cov;
    % predictive distribution (normal distribution)
    mu_pred_FG = mu_1_FG;
    cov_pred_FG = d1_FG_cov + cov_1_FG;

    % BG
    d1_BG_cov = cov(d1_BG) * (n_BG-1)/ n_BG;
    tmp3 = inv(cov_0 + (1/ n_BG)*d1_BG_cov);
    mu_1_BG = cov_0 * tmp3 * transpose(mean(d1_BG)) + (1/ n_BG) * d1_BG_cov * tmp3 *
transpose(strategy.mu0_BG);
    cov_1_BG = cov_0 * tmp3 * (1/ n_BG) * d1_BG_cov;
    % predictive distribution (normal distribution)
    mu_pred_BG = mu_1_BG;
    cov_pred_BG = d1_BG_cov + cov_1_BG;

    % Prior
    num_FG = size(d1_FG,1);
    num_BG = size(d1_BG,1);
    prior_FG = num_FG / (num_FG + num_BG);
    prior_BG = num_BG / (num_FG + num_BG);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
    % Bayes-BDR

    % BDR
    img = imread('cheetah.bmp');
    img = im2double(img);
    % Add paddle
    img = [zeros(size(img,1),2) img];
    img = [zeros(2, size(img,2)); img];
```

```
img = [img zeros(size(img,1),5)];
img = [img; zeros(5, size(img,2))];

%%% DCT
[m,n] = size(img);
Blocks = ones(m-7,n-7);
det_cov_FG = det(cov_pred_FG);
det_cov_BG = det(cov_pred_BG);
ave_tmp_FG = transpose(mu_pred_FG);
ave_tmp_BG = transpose(mu_pred_BG);
inv_tmp_FG = inv(cov_pred_FG);
inv_tmp_BG = inv(cov_pred_BG);

% predict
const_FG = ave_tmp_FG*inv_tmp_FG*transpose(ave_tmp_FG) + log(det_cov_FG) - 2*log(prior_FG);
const_BG = ave_tmp_BG*inv_tmp_BG*transpose(ave_tmp_BG) + log(det_cov_BG) - 2*log(prior_BG);

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_tmp_FG*transpose(feature);
      g_cheetah = g_cheetah - 2*feature*inv_tmp_FG*transpose(ave_tmp_FG);
      g_cheetah = g_cheetah + const_FG;
      % grass
      g_grass = g_grass + feature*inv_tmp_BG*transpose(feature);
      g_grass = g_grass - 2*feature*inv_tmp_BG*transpose(ave_tmp_BG);
      g_grass = g_grass + const_BG;
      if g_cheetah >= g_grass
         Blocks(i,j) = 0;
      end
   end
end

% save prediction
imwrite(Blocks, ['bayes_prediction_alpha_' int2str(alpha_idx) '_dataset_' int2str(dataset_idx)
'_strategy_' int2str(strategy_idx) '.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_FG;
error2_64 = (count2/count_grass_truth) * prior_BG;
total_error_64 = error1_64 + error2_64;
bayes_error = [bayes_error total_error_64];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
% ML-BDR

% ML prediction
img = imread('cheetah.bmp');
img = im2double(img);
% Add paddle
img = [zeros(size(img,1),2) img];
img = [zeros(2, size(img,2)); img];
img = [img zeros(size(img,1),5)];
img = [img; zeros(5, size(img,2))];
```

```
%%% DCT
[m,n] = size(img);
Blocks = ones(m-7,n-7);
mean_FG = mean(d1_FG);
mean_BG = mean(d1_BG);
ave_tmp_FG = mean_FG;
ave_tmp_BG = mean_BG;
inv_covFG = inv(d1_FG_cov);
inv_covBG = inv(d1_BG_cov);
DcovFG = det(d1_FG_cov);
DcovBG = det(d1_BG_cov);

%%% predict
const_FG = ave_tmp_FG*inv_covFG*transpose(ave_tmp_FG) + log(DcovFG) - 2*log(prior_FG);
const_BG = ave_tmp_BG*inv_covBG*transpose(ave_tmp_BG) + log(DcovBG) - 2*log(prior_BG);

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 + const_FG;
      % 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 + const_BG;
      if g_cheetah >= g_grass
         Blocks(i,j) = 0;
      end
   end
end

%%% save prediction
imwrite(Blocks, ['mle_prediction_alpha_' int2str(alpha_idx) '_dataset_' int2str(dataset_idx) '_strategy_'
int2str(strategy_idx) '.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_FG;
error2_64 = (count2/count_grass_truth) * prior_BG;
total_error_64 = error1_64 + error2_64;
mle_error = [mle_error total_error_64];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
% MAP-BDR

% BDR
img = imread('cheetah.bmp');
img = im2double(img);
% Add paddle
img = [zeros(size(img,1),2) img];
img = [zeros(2, size(img,2)); img];
img = [img zeros(size(img,1),5)];
img = [img; zeros(5, size(img,2))];
```

```matlab
%%% DCT
[m,n] = size(img);
Blocks = ones(m-7,n-7);
det_cov_FG = det(d1_FG_cov);
det_cov_BG = det(d1_BG_cov);
ave_tmp_FG = transpose(mu_pred_FG);
ave_tmp_BG = transpose(mu_pred_BG);

% predict
const_FG = ave_tmp_FG*inv_covFG*transpose(ave_tmp_FG) + log(det_cov_FG) - 2*log(prior_FG);
const_BG = ave_tmp_BG*inv_covBG*transpose(ave_tmp_BG) + log(det_cov_BG) - 2*log(prior_BG);

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 + const_FG;
    % 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 + const_BG;
    if g_cheetah >= g_grass
      Blocks(i,j) = 0;
    end
  end
end

% save prediction
imwrite(Blocks, ['map_prediction_alpha_' int2str(alpha_idx) '_dataset_' int2str(dataset_idx) '_strategy_'
int2str(strategy_idx) '.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_FG;
            error2_64 = (count2/count_grass_truth) * prior_BG;
            total_error_64 = error1_64 + error2_64;
            map_error = [map_error total_error_64];
        end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
        % plot
        plot(alpha,bayes_error,alpha,mle_error,alpha,map_error);
        legend('Predict','ML','MAP');
        set(gca, 'XScale', 'log');
        title('PoE vs Alpha');
        xlabel('Alpha');
        ylabel('PoE');
        saveas(gcf,['Strategy_' int2str(strategy_idx) '_dataset_' int2str(dataset_idx) '_PoEvsAlpha.png']);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
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
```