MATLAB CODE FILES

1. **max_lhood_guass_est_param.m**

```
function [ model ] = max_lhood_gauss_est_param( X, Yx,ques)
% Input  :  X - training instances, Yx - training labels
% Output :  model - contains the estimated parameters of each
%           Gaussian (means and covariance matrices)

    % Add code here for parameter estimation

    %calculate N
    N = size(Yx,1);
    N_1 = sum(Yx == 1);
    N_2 = sum(Yx == 2);
    N_3 = sum(Yx == 3);

    %calculate pi
    pi_1 = N_1/N;
    pi_2 = N_2/N;
    pi_3 = N_3/N;

    %calculate mu
    mu_1 = sum(X(Yx == 1,:))/N_1;
    mu_2 = sum(X(Yx == 2,:))/N_2;
    mu_3 = sum(X(Yx == 3,:))/N_3;

    %calculate S
    x_1 = X(Yx==1,:) - ones(size(X(Yx==1,:)))*diag(mu_1);
    x_2 = X(Yx==2,:) - ones(size(X(Yx==2,:)))*diag(mu_2);
    x_3 = X(Yx==3,:) - ones(size(X(Yx==3,:)))*diag(mu_3);
    s_1 = transpose(x_1)*x_1 / N_1;
    s_2 = transpose(x_2)*x_2 / N_2;
    s_3 = transpose(x_3)*x_3 / N_3;
    s = (N_1/N)*s_1 + (N_2/N)*s_2 + (N_3/N)*s_3;

    %calculate sigma
    switch(ques)
        case '1a'
            sigma_1 = (trace(s)/2)*eye(size(s));
            sigma_2 = sigma_1;
            sigma_3 = sigma_1;
        case '1b'
            sigma_1 = diag(diag(s));
            sigma_2 = sigma_1;
            sigma_3 = sigma_1;
        case '1c'
            sigma_1 = s;
            sigma_2 = sigma_1;
            sigma_3 = sigma_1;
        case '2a'
            sigma_1 = (trace(s_1)/2)*eye(size(s_1));
```

```
        sigma_2 = (trace(s_2)/2)*eye(size(s_2));
        sigma_3 = (trace(s_3)/2)*eye(size(s_3));

      case '2b'
        sigma_1 = diag(diag(s_1));
        sigma_2 = diag(diag(s_2));
        sigma_3 = diag(diag(s_3));
      case '2c'
        sigma_1 = s_1;
        sigma_2 = s_2;
        sigma_3 = s_3;
    end
    %calculate w
    sigma1_inv = inv(sigma_1);
    w_1 = sigma1_inv*transpose(mu_1);
    sigma2_inv = inv(sigma_2);
    w_2 = sigma2_inv*transpose(mu_2);
    sigma3_inv = inv(sigma_3);
    w_3 = sigma3_inv*transpose(mu_3);


    %calculate w0
    w0_1 = -0.5*mu_1*sigma1_inv*transpose(mu_1) + log(pi_1);
    w0_2 = -0.5*mu_2*sigma2_inv*transpose(mu_2) log(pi_2);
    w0_3 = -0.5*mu_3*sigma3_inv*transpose(mu_3) log(pi_3);


    % Change the line below and set the variable model appropriately.
    %model = [];
    model.mu_1 = mu_1;
    model.mu_2 = mu_2;
    model.mu_3 = mu_3;
    model.s_1 = s_1;
    model.s_2 = s_2;
    model.s_3 = s_3;
    model.s = s;
    model.sigma_1 = sigma_1;
    model.sigma_2 = sigma_2;
    model.sigma_3 = sigma_3;
    model.w_1 = w_1;
    model.w_2 = w_2;
    model.w_3 = w_3;
    model.w0_1 = w0_1;
    model.w0_2 = w0_2;
    model.w0_3 = w0_3;
end
```

2. **max_lhood_guass_classify.m**

```
function [ pred ] = max_lhood_gauss_classify( model, T )
% Input  : model - is the model learnt using max_lhood_gauss_est_param.m.
%          T - is the test data instances (one per row).
% Output : pred - is a vector of predicted labels (one per row)

   % Add code here for classification

   %calculate a
   a_1 = transpose(model.w_1)*T' + model.w0_1;
   a_2 = transpose(model.w_2)*T' + model.w0_2;
   a_3 = transpose(model.w_3)*T' + model.w0_3;

   %calculate probs
   exp_a1 = exp(a_1);
   exp_a2 = exp(a_2);
   exp_a3 = exp(a_3);
   exp_sum = exp_a1 + exp_a2 + exp_a3;
   %predict classes
   pred = zeros(size(T,1),1);
   p = zeros(3,1);
   for i = 1:size(T,1)
      p(1) = exp_a1(i)/exp_sum(i);
      p(2) = exp_a2(i)/exp_sum(i);
      p(3) = exp_a3(i)/exp_sum(i);

      [b j] = max(p);
      pred(i) = j;
   end

   % Change the line below and set the predictions appropriately.
   %pred = zeros(size(T,1),1);
end
```

3. multiclass_error.m

```
function [ err] = multiclass_error(y_pred,y_true)
%This function computes the multiclass error between predicted lables
%y_pred and actual lables y_true.

%y_pred - Predicted lables (m*1 vector whose each entry belongs to the set
{0,1,2,3,4,5,6,7,8,9}
%y_true - True lables (m*1 vector whose each entry belongs to the set {0,1,2,3,4,5,6,7,8,9}

%Output
%err - Fraction of data instances where y_pred and y_true do not match.

len = (length(y_pred));

err = length(find(y_pred ~= y_true))/len;

end
```

4. contour_plot.m

```
function contour_plot(mu, sigma, linespec)
% Input  : mu - is the mean of a 2-D Gaussian.
%          sigma - is the 2x2 covariance matrix of the Gaussian.
%          linespec - use this to specify the color of the plot,
%          e.g. 'r', 'g', 'b', etc.

% Output : Contour plot for the Gaussian
% Tip    : Use "hold on;" and call this three times (one for each Gaussian).
x1range=0:0.1:4;
x2range=0:0.1:4;
[X1 X2] = meshgrid(x1range, x2range);

Z = zeros(length(x2range), length(x1range));
for n1 = 1:length(x2range)
    for n2 = 1:length(x1range)
        Z(n1,n2) = mvnpdf([X1(n1,n2) X2(n1,n2)], mu, sigma);
    end
end
contour(X1, X2, Z, 2, linespec);
axis square;
```

5. **decision_boundary.m**

```
function decision_boundary( model )
% Input  : model - is the model learnt using max_lhood_gauss_est_param.m.
% Output : Decision boundary plot for the model

x1range=0:0.02:3;
x2range=0:0.02:4;
M = length(x1range);
N = length(x2range);
[X1 X2] = meshgrid(x1range, x2range);
X1 = reshape(X1, M*N, 1);
X2 = reshape(X2, M*N, 1);
X = [X1 X2];

pred = max_lhood_gauss_classify(model, X);

hold on;
c1 = pred==1;
c2 = pred==2;
c3 = pred==3;

plot(X(c1,1), X(c1,2),'r.');
plot(X(c2,1), X(c2,2),'g.');
plot(X(c3,1), X(c3,2),'b.');

end
```