Assignment I

1.2 Train class conditional Gaussians

Source code:

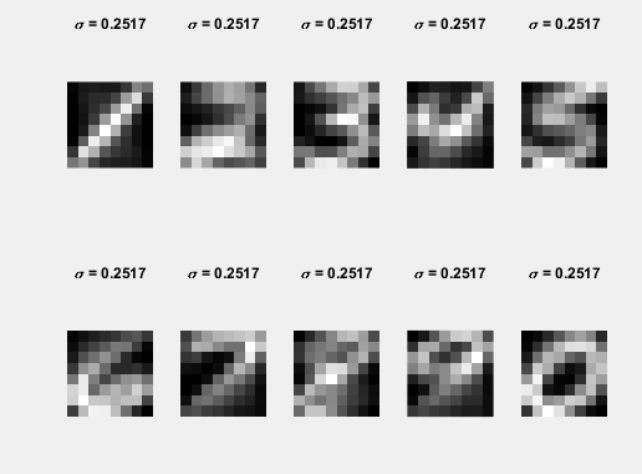
```
% Initial setup
clear all
clc
load a1digits.mat

% Define all global values
% D is the a real valued vector (8 by 8 in size)
D = size(digits_train,1);
% M is the number of all training data points (700 training cases)
M = size(digits_train,2);
% K is the true label of each classes (from 1 to 10)
K = size(digits_train,3);

% 1.2 Training class conditional Gaussians with independent features
% Main goal is to calculate sigma_square and miu_k
% calculate sigma_square and miu_k by using the equation
val =  repmat(mean(digits_train,2),1,M,1);
power = ((digits_train - val).^2);
sigma_square = sum(sum(sum(power))) / (M*D*K);
sigma = sqrt(sigma_square);
miu_k = mean(digits_train,2);

% Plot the figure
figure
for a = 1:K
    subplot(2,5,a)
    imagesc(reshape(miu_k(:,1,a),8,8)'); axis equal; axis off; colormap gray;
    name = [ '{\sigma} = ',num2str(sigma)];
    title(name)
end
```

Output:



2.0 Training Naïve Bayes Classifiers

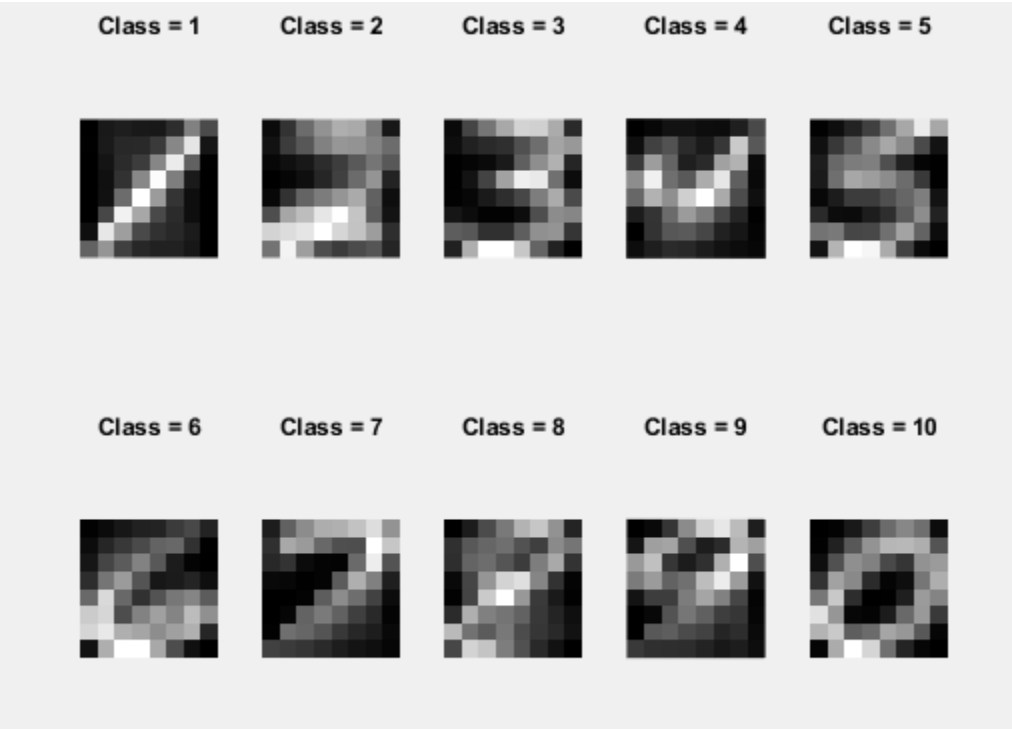
Source code:

```
% 2 Training Naive Bayes Classifiers
Naive_Bayes = digits_train;
for a = 1:K
    for b = 1:M
        for c = 1:D
            if Naive_Bayes(c,b,a)> 0.5;
                Naive_Bayes(c,b,a)=1;
            else
                Naive_Bayes(c,b,a) = 0;
            end
        end
    end
end

% Apply the Naive Bayes equation
total = sum(Naive_Bayes,2) / M;
total = reshape(total,D,K);

% Plot the figure
figure
for a = 1:K
    subplot(2,5,a)
    imagesc(reshape(total(:,a),8,8)'); axis equal; axis off;colormap gray;
    title(['Class = ',num2str(a)]);
end
```

Output:



3.0 Test Performance

1. Conditional Gaussians

Source code:

```
% 3 Test Performance
% Conditional Gaussians
for a = 1:10
    for b = 1:400
        for c = 1:10
            val = sum((digits_test(:,b,a) - miu_k(:,1,c)).^2,1);
            pxCk(c) = (2*pi*sigma)^(-D/2).*exp(-1/(2*sigma)*val);
        end
        pCkx = pxCk*0.1./(1/400);
        [~,label(b)] = max(pCkx);
    end
    Gaussians_error(a) = 400-sum(label==a);
    Gaussians_error_rate(a) = (Gaussians_error(a)/400)*100;
end
Gaussians_overall_error_rate = sum(Gaussians_error)/4000*100
table = [Gaussians_error;Gaussians_error_rate];
fprintf('Conditional Gaussians:');
Output = [Gaussians_error;Gaussians_error_rate];
Output_1 =
array2table(Output,'VariableNames',{'1','2','3','4','5','6','7','8','9','10'}
,'RowNames',{'Gaussians_error','Gaussians_error_rate'});
Output_1
```

Output:

```
Gaussians_overall_error_rate =

   18.0250

Conditional Gaussians:
Output_1 =

  2×10 table
```

	1	2	3	4	5	6	7	8	9	10
Gaussians_error	69	81	63	61	68	44	63	109	110	53
Gaussians_error_rate	17.25	20.25	15.75	15.25	17	11	15.75	27.25	27.5	13.25

2. Naïve Bayes

Source code:

```
% Naive Bayes
for a = 1:10
```

```
    for b = 1:400
        for c = 1:10
            d = digits_test(:,b,a);
            for e = 1:64
                if d(e,1) >0.5
                    d(e,1)=1;
                else
                    d(e,1) = 0;
                end
            end
            eta = total(:,c);
            pbCketa(c) = 1;
            for x = 1:64
                if d(x) ==1
                    pbCketa(c) = pbCketa(c) * eta(x);
                else
                    pbCketa(c) = pbCketa(c) * (1-eta(x));
                end
            end
        end
        pCkbeta = pbCketa./sum(pbCketa);
        [~,lable(b)] = max(pCkbeta);
    end
    Naive_error(a) = 400-sum(lable==a);
    Naive_error_rate(a) = ((400-sum(lable==a))/400)*100;
end
Naive_overall_error_rate = sum(Naive_error)/4000*100
fprintf('Naive Bayes:');
Output_3 = [Naive_error;Naive_error_rate];
Output_2 =
array2table(Output_3,'VariableNames',{'1','2','3','4','5','6','7','8','9','10
'},'RowNames',{'Naive_error','Naive_error_rate'});
Output_2
```

Output:

```
Naive_overall_error_rate =

   23.4750

Naive Bayes:
Output_2 =

  2×10 table
```

	1	2	3	4	5	6	7	8	9	10
Naive_error	87	104	91	85	111	60	89	121	133	58
Naive_error_rate	21.75	26	22.75	21.25	27.75	15	22.25	30.25	33.25	14.5

method/number	1	2	3	4	5	6	7	8	9	10	Overall rate
Gaussian error	69	81	63	61	68	44	63	109	110	53	18.0250
Naïve error	87	104	91	85	111	60	89	121	133	58	23.4750