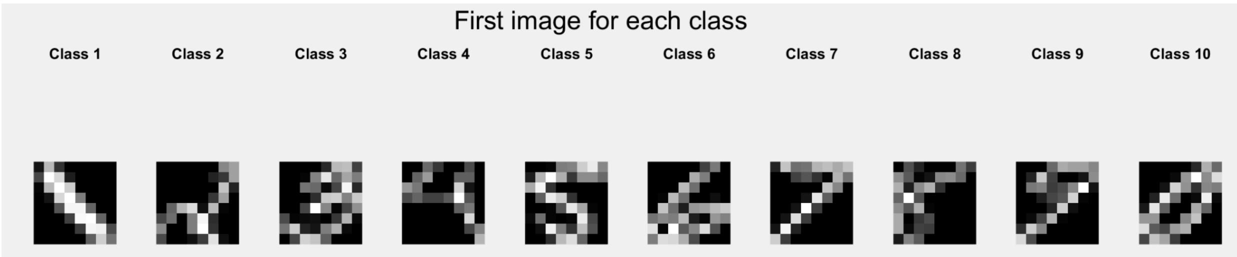
1 Exploring the Dataset

The first images of each of the classes were plotted in the image below.

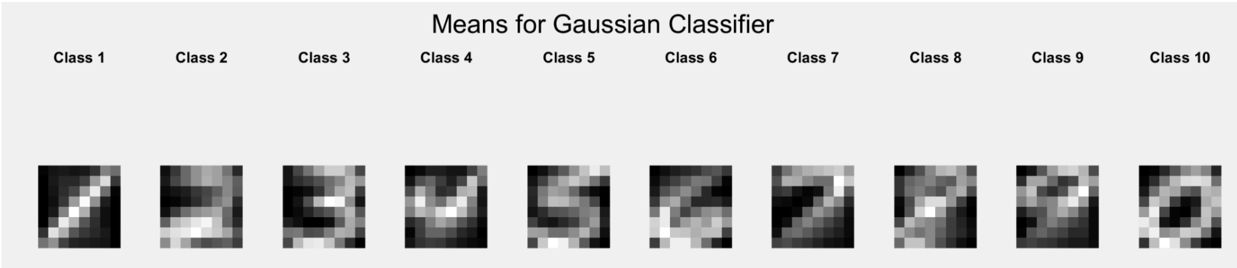


Code:

```
clear
load assignment1.mat
% create consistent plotting settings
figure_settings = struct(...
   'Position', [100 100 1200 300], ...
   'ColorMap', gray, ...
   'SubplotDims', [1 10]);
% visualize first digit of each class
figure('Position', figure_settings.Position);
for k = 1:10
   subplot(figure_settings.SubplotDims(1), figure_settings.SubplotDims(2), k);
   imagesc(reshape(digits_train(:,1,k), 8, 8)');
   colormap(figure_settings.ColorMap);
   axis equal;
   axis off;
   title(['Class ' num2str(k)]);
end
sgtitle('First image for each class');
```

2 Training Gaussian Classifiers

The image below visualizes the means for the 10 classes.



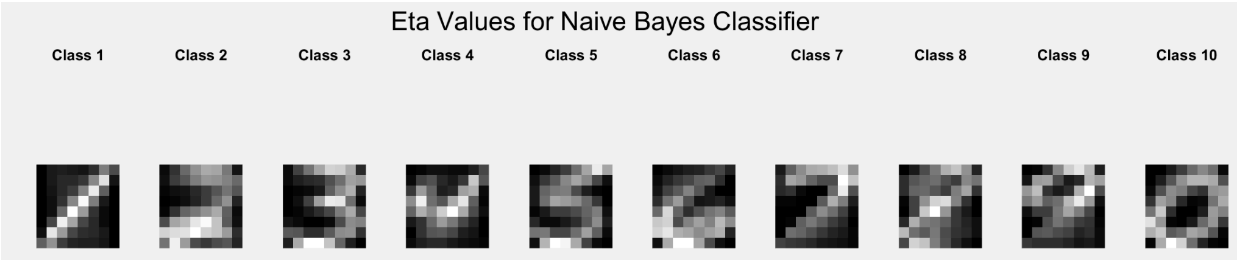
The computed variance (sigma squared) is 0.063351

Code:

```
D=64;
num_classes=10;
% calculate means
means = reshape(mean(digits_train, 2), [64, 10]); %the mean function gives
64x1x10 but want 64x10
% calculate shared variance (sigma^2) with equation (4)
sigma_squared = 0;
M = size(digits_train, 2) * size(digits_train, 3);  % total number of training
samples 700x10 in this case
% calculate numerator of sigma^2
for k = 1:10
   for j = 1:size(digits_train,2)
       for i = 1:D
           sigma_squared = sigma_squared + (digits_train(i,j,k) -
means(i,k))^2;
       end
   end
end
% complete sigma^2 calculation
sigma_squared = sigma_squared / (D*M);
fprintf('Variance = %.6f\n', sigma_squared)
% plot means (gaussian classifier)
figure('Position', figure_settings.Position);
for k = 1:10
   subplot(figure_settings.SubplotDims(1), figure_settings.SubplotDims(2), k);
   imagesc(reshape(means(:,k), 8, 8)');
   colormap(figure_settings.ColorMap);
   axis equal;
   axis off;
   title(['Class ' num2str(k)]);
end
sgtitle('Means for Gaussian Classifier');
```

3 Training Naive Bayes Classifiers

The image below visualizes the eta values for each of the classes.



Code:

```
% convert training data to binary using threshold of 0.5
digits_train_binary = (digits_train > 0.5);
% calculate eta
eta = reshape(mean(digits_train_binary, 2), [64, 10]); %the mean function gives 64x1x10 but want 64x10
% plot eta values (naive bayes classifier)
figure('Position', figure_settings.Position);
for k = 1:10
   subplot(figure_settings.SubplotDims(1), figure_settings.SubplotDims(2), k);
   imagesc(reshape(eta(:,k), 8, 8)');
   colormap(figure_settings.ColorMap);
   axis equal;
   axis off;
   title(['Class ' num2str(k)]);
end
sgtitle('Eta Values for Naive Bayes Classifier');
```

4 Test Performance

The table below lists the number of errors made by the gaussian classifier and naive bayes classifier in each class of the test set and the overall error rates.

Class	1	2	3	4	5	6	7	8	9	10	Overall error rates:
Gaussian	69	81	63	61	68	44	63	109	110	53	18.02%
Naive Bayes	87	104	91	85	111	60	89	121	133	58	23.47%

The overall error rates for both classifiers:

- Gaussian Classifier: 18.02%
- Naive Bayes Classifier: 23.47%

The gaussian classifier achieved a lower overall error rate than naive bayes as well as the number of errors for each of the classes.

Code:

```
%test%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% variables to store the errors per class
gaussian_error_count=zeros(1, 10);
naive_bayes_error_count= zeros(1, 10);
num_test_per_class = size(digits_test, 2);%400 in this case
for k = 1:10
   for j = 1:num_test_per_class
       test_image = digits_test(:,j,k);
       % test gaussian classifier
       gaussian_pred = gaussian_test(test_image, means, sigma_squared);
       if gaussian_pred ~= (k)
           gaussian_error_count(k) = gaussian_error_count(k) + 1;
       end

       % test naive bayes classifier
       naive_bayes_pred = naive_bayes_test(test_image, eta);
       if naive_bayes_pred ~= (k)
           naive_bayes_error_count(k) = naive_bayes_error_count(k) + 1;
       end
   end
end
% calculate overall error rates
gaussian_error_rate = sum(gaussian_error_count) / (num_test_per_class * 10);
```

```
naive_bayes_error_rate = sum(naive_bayes_error_count) / (num_test_per_class *
10);
% display results
fprintf('\nError counts per class:\n');
fprintf('Class\tGaussian\tNaive Bayes\n');
for k = 1:10
   fprintf('%d\t%d\t\t%d\n', k, gaussian_error_count(k),
naive_bayes_error_count(k));
end
fprintf('\nOverall error rates:\n');
fprintf('Gaussian Classifier: %.2f%%\n', gaussian_error_rate * 100);
fprintf('Naive Bayes Classifier: %.2f%%\n', naive_bayes_error_rate * 100);
%functions%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% predict a class of an image with gaussian
function predicted_class = gaussian_test(test_image, means, sigma_squared)
   num_classes=10;
   D=64;
   probs = zeros(10, 1);
   for k = 1:num_classes
       % equation (1)
       diff = test_image - means(:,k);
       probs(k) = (2*pi*sigma_squared)^(-D/2)  *exp(
-sum(diff.^2)/(2*sigma_squared));
   end
   [~, predicted_class] = max(probs);

end
% predict a class of an image with naive bayes
function predicted_class = naive_bayes_test(test_image, eta)
   num_classes=10;
   binary_test = test_image > 0.5;
   probs = ones(10, 1);
   for k = 1:num_classes
       % equation  (7)
       for i = 1:64
           if(binary_test(i))
               probs(k)=probs(k)*eta(i,k);
           else
               probs(k)=probs(k)*(1-eta(i,k));
           end
       end
   end
   [~, predicted_class] = max(probs);

end
```

Full code

```
clear
load assignment1.mat
% create consistent plotting settings
figure_settings = struct(...
   'Position', [100 100 1200 300], ...
   'ColorMap', gray, ...
   'SubplotDims', [1 10]);
% visualize first digit of each class
figure('Position', figure_settings.Position);
for k = 1:10
   subplot(figure_settings.SubplotDims(1), figure_settings.SubplotDims(2), k);
   imagesc(reshape(digits_train(:,1,k), 8, 8)');
   colormap(figure_settings.ColorMap);
   axis equal;
   axis off;
   title(['Class ' num2str(k)]);
end
sgtitle('First image for each class');
%gaussian classifier%%%%%%%%%%%%%%%%%%
%{
p(x|C_k) = (2pi*sigma^2)^(-D/2) * exp{-(1/2*sigma^2 * sum(x_i-mu_ki)^2}
D is 64
X is each image (x1,x2,xd)
p(C_k) = a_k
%}
D=64;
num_classes=10;
% calculate means
means = reshape(mean(digits_train, 2), [64, 10]); %the mean function gives
64x1x10 but want 64x10
% calculate shared variance (sigma^2) with equation (4)
sigma_squared = 0;
M = size(digits_train, 2) * size(digits_train, 3);  % total number of training
samples 700x10 in this case
% calculate numerator of sigma^2
for k = 1:10
   for j = 1:size(digits_train,2)
       for i = 1:D
           sigma_squared = sigma_squared + (digits_train(i,j,k) -
means(i,k))^2;
       end
   end
end
% complete sigma^2 calculation
sigma_squared = sigma_squared / (D*M);
fprintf('Variance = %.6f\n', sigma_squared)
% plot means (gaussian classifier)
```

```matlab
figure('Position', figure_settings.Position);
for k = 1:10
   subplot(figure_settings.SubplotDims(1), figure_settings.SubplotDims(2), k);
   imagesc(reshape(means(:,k), 8, 8)');
   colormap(figure_settings.ColorMap);
   axis equal;
   axis off;
   title(['Class ' num2str(k)]);
end
sgtitle('Means for Gaussian Classifier');
%naive bayes classifier%%%%%%%%%%%%%%%%%%%%%%%%%
% convert training data to binary using threshold of 0.5
digits_train_binary = (digits_train > 0.5);
% calculate eta
eta = reshape(mean(digits_train_binary, 2), [64, 10]); %the mean function gives
64x1x10 but want 64x10
% plot eta values (naive bayes classifier)
figure('Position', figure_settings.Position);
for k = 1:10
   subplot(figure_settings.SubplotDims(1), figure_settings.SubplotDims(2), k);
   imagesc(reshape(eta(:,k), 8, 8)');
   colormap(figure_settings.ColorMap);
   axis equal;
   axis off;
   title(['Class ' num2str(k)]);
end
sgtitle('Eta Values for Naive Bayes Classifier');
%test%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% variables to store the errors per class
gaussian_error_count=zeros(1, 10);
naive_bayes_error_count= zeros(1, 10);
num_test_per_class = size(digits_test, 2);%400 in this case
for k = 1:10
   for j = 1:num_test_per_class
       test_image = digits_test(:,j,k);
       % test gaussian classifier
       gaussian_pred = gaussian_test(test_image, means, sigma_squared);
       if gaussian_pred ~= (k)
           gaussian_error_count(k) = gaussian_error_count(k) + 1;
       end

       % test naive bayes classifier
       naive_bayes_pred = naive_bayes_test(test_image, eta);
       if naive_bayes_pred ~= (k)
           naive_bayes_error_count(k) = naive_bayes_error_count(k) + 1;
       end
   end
end
% calculate overall error rates
```

```
gaussian_error_rate = sum(gaussian_error_count) / (num_test_per_class * 10);
naive_bayes_error_rate = sum(naive_bayes_error_count) / (num_test_per_class *
10);
% display results
fprintf('\nError counts per class:\n');
fprintf('Class\tGaussian\tNaive Bayes\n');
for k = 1:10
   fprintf('%d\t%d\t\t%d\n', k, gaussian_error_count(k),
naive_bayes_error_count(k));
end
fprintf('\nOverall error rates:\n');
fprintf('Gaussian Classifier: %.2f%%\n', gaussian_error_rate * 100);
fprintf('Naive Bayes Classifier: %.2f%%\n', naive_bayes_error_rate * 100);
%functions%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% predict a class of an image with gaussian
function predicted_class = gaussian_test(test_image, means, sigma_squared)
   num_classes=10;
   D=64;
   probs = zeros(10, 1);
   for k = 1:num_classes
       % equation (1)
       diff = test_image - means(:,k);
       probs(k) = (2*pi*sigma_squared)^(-D/2)  *exp(
-sum(diff.^2)/(2*sigma_squared));
   end
   [~, predicted_class] = max(probs);

end
% predict a class of an image with naive bayes
function predicted_class = naive_bayes_test(test_image, eta)
   num_classes=10;
   binary_test = test_image > 0.5;
   probs = ones(10, 1);
   for k = 1:num_classes
       % equation  (7)
       for i = 1:64
           if(binary_test(i))
               probs(k)=probs(k)*eta(i,k);
           else
               probs(k)=probs(k)*(1-eta(i,k));
           end
       end
   end
   [~, predicted_class] = max(probs);

end
```