

Queen's University – Smith Engineering

ELEC 425 – Assignment 1 Fall 2025

Submitted by:

Erhowvosere Otubu – Student Number: 20293052

Date Submitted: Wednesday, October 8<sup>th</sup>, 2025

## Table of Contents

Training Gaussian Classifiers .....	1
Training Naïve Bayes Classifiers .....	3
Test Performance .....	4

## Table of Figures

Figure 1: Plot for 10 means side by side, each showing an 8 x 8 image .....	1
Figure 2: Source code used to calculate shared variance and mean calculations.....	1
Figure 3: Value of the shared variance from the command window .....	2
Figure 4: Plat for 10 estimates side by side, each showing an 8 x 8 image .....	3
Figure 5: Source code used calculate the parameter estimates.....	3
Figure 6: 2-by-10 table showing the number of errors each of the two classifiers make on each of the 10 digits including the overall error rate for each classifier .....	4
Figure 7: Source code used for testing Gaussian Classifier performance.....	4
Figure 8: Source code used for testing Naive Bayes Classifier performance.....	5

## Training Gaussian Classifiers

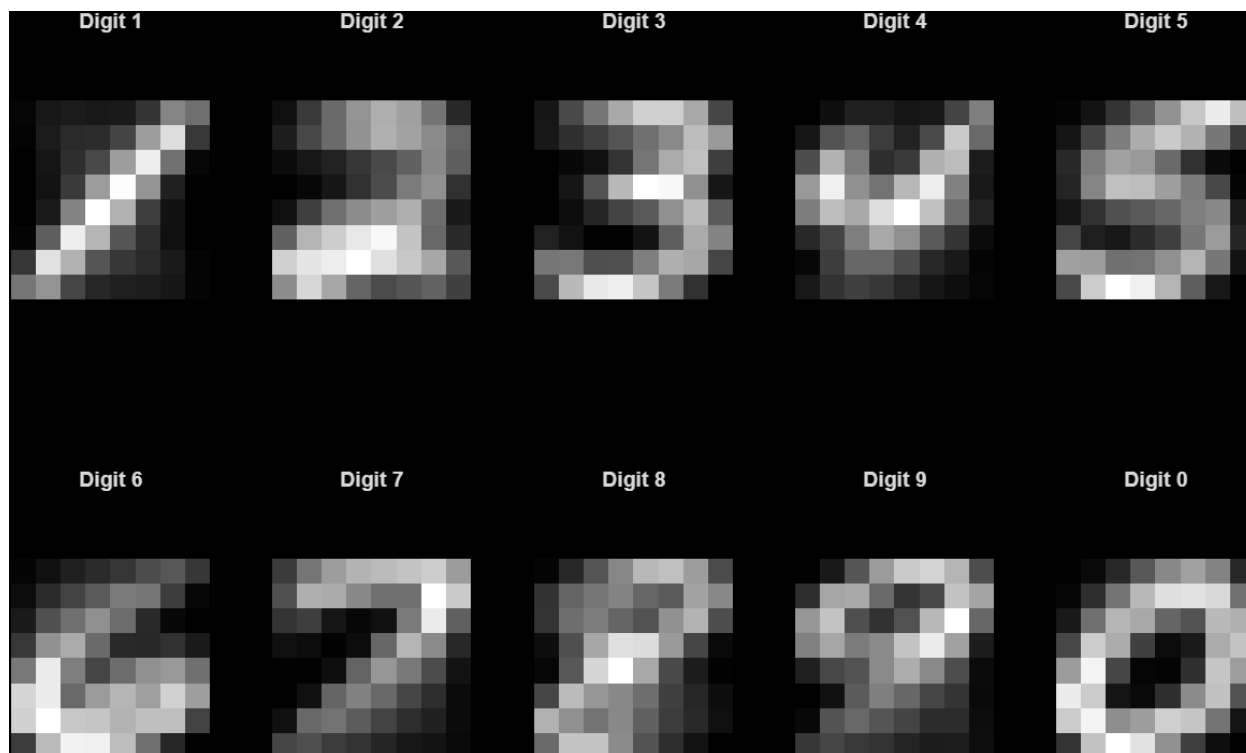


Figure 1: Plot for 10 means side by side, each showing an 8 x 8 image

```
D = 64; % Number of features
K = 10; % Number of class-conditional Gaussians
mk = 700; % Number of training images
mt = 400; % Number of test images

% Create a D x K (64 x 10) matrix of zeros to store means
mu = zeros(D, K);

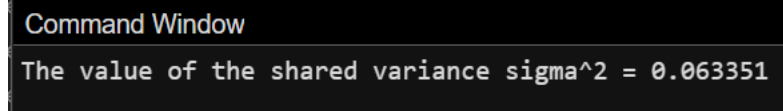
% Initialize matrix that will store Gaussian Classifier errors
errors_gaussian = zeros(1, K);

for k = 1:K
    mu(:,k) = mean(digits_train(:, :, k), 2);
end

M = K * mk; % Number of all training data points over all K classes
sigma_sq = 0;

for k = 1:K
    diff = digits_train(:, :, k) - mu(:, k);
    sigma_sq = sigma_sq + sum(diff(:).^2);
end
sigma_sq = sigma_sq / (D * M);
```

Figure 2: Source code used to calculate shared variance and mean calculations



Command Window

The value of the shared variance  $\sigma^2 = 0.063351$

*Figure 3: Value of the shared variance from the command window*

## Training Naïve Bayes Classifiers

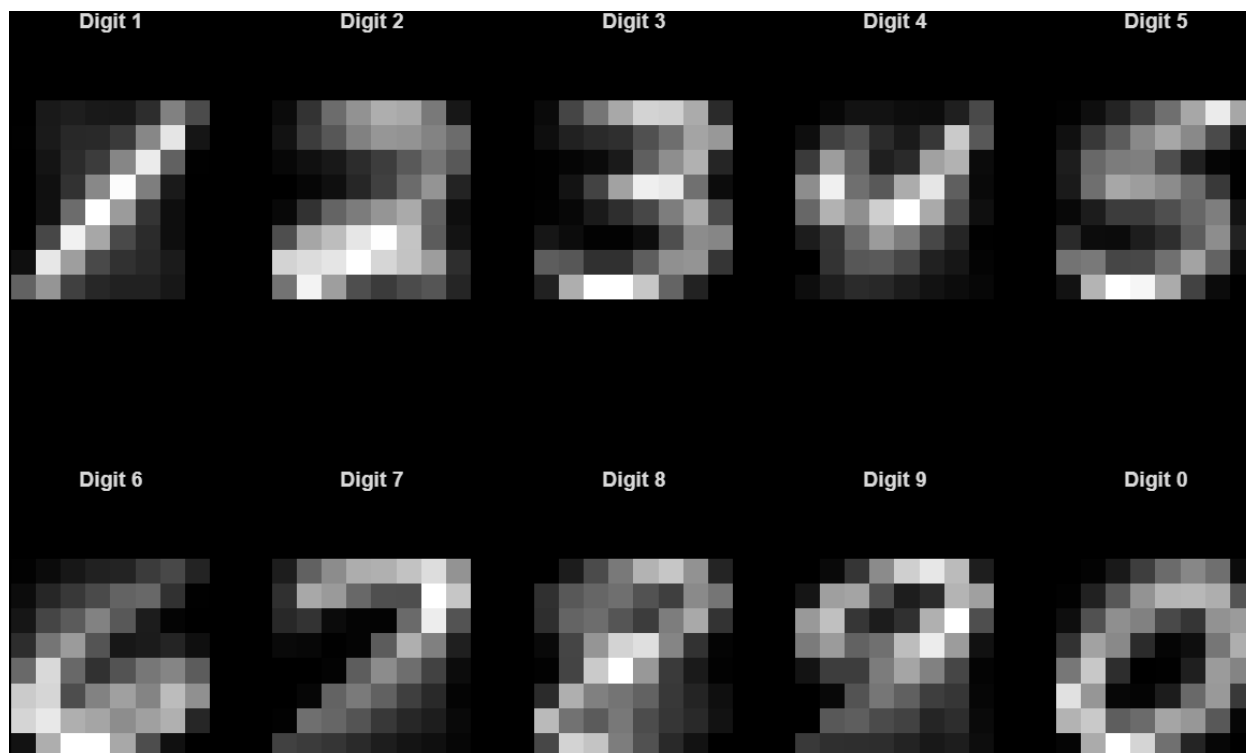


Figure 4: Plot for 10 estimates side by side, each showing an 8 x 8 image

```
D = 64; % Number of features
K = 10; % Number of class-conditional Gaussians
mk = 700; % Number of training images
mt = 400; % Number of test images

% Converting the real-valued features into binary features for training
binary_features_train = digits_train > 0.5;
parameters = zeros(D, K);

for k = 1:K
    % Since the data are 0/1, the mean is exactly the probability a
    % pixel 'On' for that digit
    parameters(:, k) = mean(binary_features_train(:, :, k), 2);
end
```

Figure 5: Source code used calculate the parameter estimates

## Test Performance

Error counts per digit:										
	0	1	2	3	4	5	6	7	8	9
	—	—	—	—	—	—	—	—	—	—
Gaussian	69	81	63	61	68	44	63	109	110	53
Naive Bayes	87	104	91	85	111	60	89	121	133	58

Gaussian total error rate: 0.1802  
Naive Bayes total error rate: 0.2347

Figure 6: 2-by-10 table showing the number of errors each of the two classifiers make on each of the 10 digits including the overall error rate for each classifier

```
%-----Test Section-----
for k = 1:K
    X_test = digits_test(:, :, k);
    pbxCk = zeros(K, mt);
    % Loop through all possible classes
    for c = 1:K
        % difference between each test image and the mean of class c
        diff = X_test - mu(:, c);
        % square the differences and sum across all 64 features
        diff_sq = sum(diff.^2, 1);
        % Applying Gaussian formula
        pbxCk(c, :) = (2*pi*sigma_sq)^(-D/2) .* exp( -diff_sq / (2*sigma_sq) );
    end
    % Finding the class with the highest likelihood
    [~, predicted] = max(pbxCk, [], 1);
    % Counting how many test images were misclassified
    errors_gaussian(k) = sum(predicted ~= k);
end

total_errors_gaussian = sum(errors_gaussian);
error_rate_gaussian = total_errors_gaussian / (K * mt);
```

Figure 7: Source code used for testing Gaussian Classifier performance

```

%-----Test Section-----
% Converting the real-valued features into binary features for testing
binary_features_test = digits_test > 0.5;
% Initialize matrix that will store Native Bayes Classifier errors
errors_nb = zeros(1, K);

for k = 1:K
    B_test = binary_features_test(:, :, k);
    pbCk = zeros(K, mt);
    % Loop through all possible classes
    for c = 1:K
        pb1Ck = parameters(:, c);
        pb0Ck = 1 - pb1Ck;
        % Applying Native Bayes formula given in assignment
        pbCk(c, :) = prod((pb1Ck.^ B_test) .* (pb0Ck.^ (1 - B_test)), 1);
    end
    % Finding the class with the highest likelihood
    [~, predicted] = max(pbCk, [], 1);
    % Counting how many test images were misclassified
    errors_nb(k) = sum(predicted ~= k);
end

total_errors = sum(errors_nb);
error_rate_nb = total_errors / (K * mt);

```

Figure 8: Source code used for testing Naive Bayes Classifier performance