Indian Institute of Technology, Guwahati

EE657 : Pattern Recognition and Machine Learning

Assignment 1 :

Bayesian Classifier based Character Recognition System

Instructor : Dr. Suresh Sundaram, IIT Guwahati

Name : Akshat Bordia
Roll No. : 11010848
Branch : Electronics and Electrical Engineering
Email ID : a.bordia@iitg.ernet.in

Contents

Note: Section 1, 2, 3 are written in order to check understanding of topic and given problem. One might skip those sections and directly go through section 4 to 8.

1. Problem Statement and Objective:

In this assignment, our objective is to create a handwritten character recognition system for 3 categories of characters: ‘e’, ‘c’ and ‘i’. This classification is done based on **Bayesian Decision Theory**.

Following are the important aspects of given problem statement:

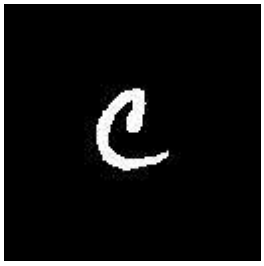
1. **Training Data Set**: To train our classifier, we are supplied training data set which contains 200 images of size 128*128 for each of given category. These images contain various variations of each handwritten characters. Based on these images and extracting distinguishing features from these images, character recognition system is modelled.

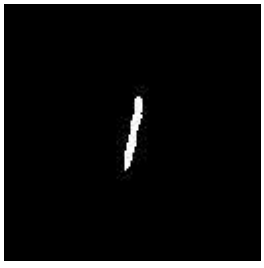
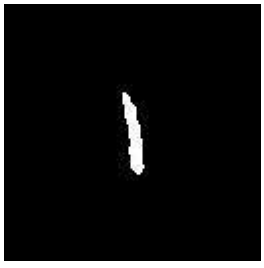
2. **Testing Data Set:** After creating character recognition system, we need to test our classifier whether it is working successfully or not, what is the accuracy of character recognition system and other such performance parameters. For this, we are given 300 images of size 128*128.

3. **Sample Distribution and Methods to be used:** The samples are assumed to be generated from a multi-dimensional Gaussian distribution, having class specific mean vector μ. **Bayesian Decision Theory** is used to create generative Bayesian classifiers. For estimation of mean and variance from the training data, we use **Maximum Likelihood Method.**

Few examples of training images:





2. Bayesian Decision Theory

The fact that only **Priors** are not sufficient to make a decision about **State of Nature** of an object can be understood very easily. Prior information will always support one class or state of nature more than others and that will defeat the whole purpose of classifier. Thus we need to incorporate certain attributes of various state of nature called as **Features** before taking a decision about their class or state of nature. Bayesian Decision Theory provides a simple method to incorporate feature knowledge in terms of **Class Condition Probability.** The foundation of this theory essentially lies in Bayes' Formula for conditional probability:

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)},$$

$$\text{Posterior} = \frac{Likelihood * Prior}{evidence}$$

Here:

$P(\omega_j \mid x)$: Posterior Probability

$p(x \mid \omega_j)$: Class Conditional Probability Density Function, Distribution of *feature x* for a given *state of nature* ω_j.

$P(\omega_j)$: Prior Probability

$p(x)$: Evidence

Based on Posterior value, it is decided that the particular object belongs to which class or state of nature. For a given feature x, the object is assigned to the class for which Posterior is maximum. This is called Bayes' Decision Rule.

Discriminant Function: One of the easiest ways to represent a classifier and take decision is in terms of Discriminant Function. The classifier assigns a feature vector X to class (i) if:

$$g_i(x) > g_j(x) \quad \text{for all } i \neq j$$

Thus, the classifier is viewed as a network or machine that computes *c* discriminant functions and selects the category corresponding to the largest discriminant.

3. Bayesian Decision Theory in Character Recognition

To create a Bayesian Classifier for the given character classes, we need following:

1. **Prior**: Priors are assumed to be equal as we have total 300 test images which contain 100 images of each character class.
   Thus prior = 1/3= equal for each class.
   Here prior is equal in each class, hence it does not contribute to our decision and it is solely based on *Class Condition Probability Density.*

2. **Class Condition Probability Density:** As given in assignment problem statement, each of the classes is assumed to come from **multi-dimensional Gaussian distribution**. These Distributions are estimated from training data using **Maximum Likelihood Method.**
   The general Multi Variate Gaussian/Normal Density can be formulated as:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

   Here X is a d-component column vector; μ is the d-component mean vector and ∑ is d-by-d covariance matrix. μ And ∑ are given respectively:

$$\boldsymbol{\mu} \equiv \mathcal{E}[\mathbf{x}] = \int \mathbf{x} p(\mathbf{x})\, d\mathbf{x}$$

$$\boldsymbol{\Sigma} \equiv \mathcal{E}[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^t] = \int (\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^t p(\mathbf{x})\, d\mathbf{x}$$

So In order to determine the multivariate Gaussian distribution, we need to determine μ and ∑ matrix.

4. Analysis and Formulation of Problems

1. **Selection of Feature**: First of all we need to choose our feature vectors for which we will be estimating distribution. In these images, an elementary feature is pixel intensity values. Thus to create feature vectors, we need to read images and store the pixel values in an array. Here we take a 128-by-128 image or resized 32-by-32 image and create a column vector of 128*128-by-1 or 32*32-by-1 accordingly. Thus for each image in each class, we have a feature column vector.

   Certainly we can explore better features to get more accuracy and develop better model. But as an initial project in the Pattern Classification, this is good enough to use this trivial feature. Other feature to achieve high accuracy can be explored in bonus section of this assignment.

2. **Calculation of Mean Vector:** For each image of a class, we need to create a feature vector as explained in above section. Now we need to determine mean vector for this class. For this, we use **Maximum Likelihood** method which gives us :

$$\mu = \frac{\sum Xi}{n}$$

   Where Xi's are the different feature vectors of a particular class and n is the no. of samples of a particular class.
   In the similar way, we determine the mean for all three classes.

3. **Calculation of Covariance Matrix:** The third parameter we need to know in order to model the multi Variate Gaussian distribution of our data is ***Covariance Matrix.*** For calculating covariance matrix, we follow the conditions given in problem statement :

   (i) **The samples of a given character class are modelled by a separate covariance matrix**.
   In this case, we will have three covariance matrices, each for a state of nature. For a given class, we can write covariance matrix using maximum likelihood:

$$\hat{\Sigma} = \frac{1}{n}\sum_{k=1}^{n}(\mathbf{x}_k - \hat{\mu})(\mathbf{x}_k - \hat{\mu})^t.$$

   Where μ cap is sample mean estimated using maximum likelihood. (Described earlier) Thus we will calculate three covariance matrices, one for each state of nature.

(ii) The samples across all the character classes are pooled together to generate a common non diagonal covariance matrix.

In this case, same formula mentioned above is used. But here all the data i.e. 600 images which includes 200 for each category are pooled together and a common covariance matrix is generated.

(iii) The samples of a given character class are separately modelled by a diagonal covariance matrix. The diagonal entries of the matrix correspond to the variances of the individual features. The features are assumed to be independent- hence their Cross variances are forced to zero.

In this scenario, after obtaining each covariance matrix in **part (i) ,** their non-diagonal elements are forced to be zero and hence making the features independent.

(iv) The samples across all the character classes are pooled to generate a common diagonal covariance matrix. The diagonal entries correspond to the variances of the individual features, which are considered to be independent.

Following part (ii), we keep the non-diagonal elements =0 in this case and a common diagonal covariance matrix is created.

(v) The covariance matrix of each class is forced to be spherical.
In this case, we take the average of diagonal entries in diagonal matrix of each class and define the covariance matrix as:

$$\sum = \{(\text{sigma}) \text{ ^2}\}*I$$

---

4. Defining Discriminant Function and Decision Surface :

(I) The samples of a given character class are modelled by a separate covariance matrix.
Discriminant function, when each class has a separate covariance matrix is given by following equation:

$$g_i(\mathbf{x}) = \mathbf{x}^t \mathbf{W}_i \mathbf{x} + \mathbf{w}_i^t \mathbf{x} + w_{i0},$$

$$\mathbf{W}_i = -\frac{1}{2}\boldsymbol{\Sigma}_i^{-1},$$

$$\mathbf{w}_i = \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i$$

$$w_{i0} = -\frac{1}{2}\boldsymbol{\mu}_i^t \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i - \frac{1}{2}\ln\,|\boldsymbol{\Sigma}_i| + \ln\,P(\omega_i).$$

X is assigned to the category for which the discriminant function has maximum value. Here we can ignore prior terms as it is equal for all the classes.

(II) The samples across all the character classes are pooled together to generate a common non diagonal covariance matrix.

In this case, we have same covariance matrix for all the classes. Hence the above expression is simplified and we get the discriminant function as:

$$g_i(\mathbf{x}) = \mathbf{w}_i^t \mathbf{x} + w_{i0},$$

$$\mathbf{w}_i = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i$$

$$w_{i0} = -\frac{1}{2}\boldsymbol{\mu}_i^t \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i + \ln\,P(\omega_i).$$

(III) The samples of a given character class are separately modelled by a diagonal covariance matrix. The diagonal entries of the matrix correspond to the variances of the individual features. The features are assumed to be independent- hence their Cross variances are forced to zero.

Here we have three different covariance matrices for the three states of natures, Hence the expression for discriminant function will same as part (I)

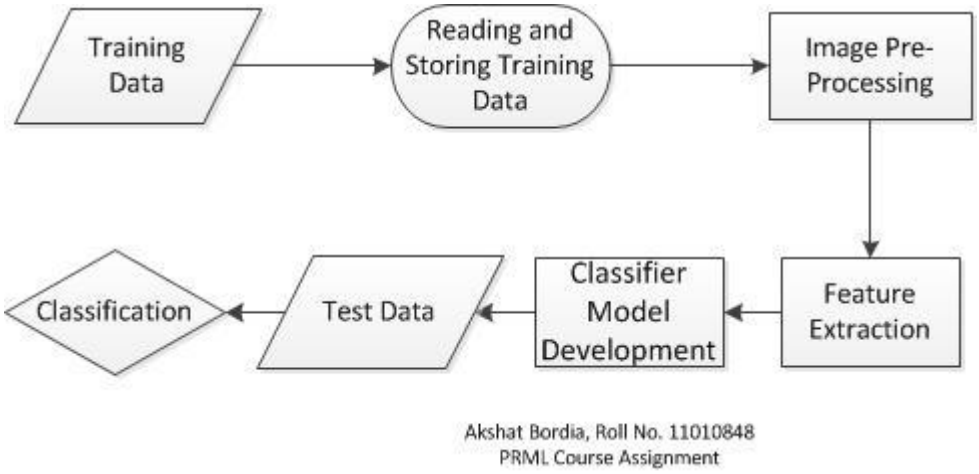
(IV) **The samples across all the character classes are pooled to generate a common diagonal covariance matrix. The diagonal entries correspond to the variances of the individual features, which are considered to be independent.**
Here discriminant function will be same as in part (II) and will be calculated using new diagonal matrix in this case.

(V) **The covariance matrix of each class is forced to be spherical.**
Here we have three different matrices with spherical distributions of data. Here also expression for discriminant function will be same as in Part (I)

5. Implementation and Code Description

This algorithm for character recognition was implemented in MATLAB2012 (b) using commands of MATLAB Image Processing Toolbox (IPT).

Flow of code can be explained by this block diagram:



Methods and Commands used in every step are explained below:

1. Reading and Storing Training Data :

We have 200 training images for each class of characters. Before we start proceeding with problem, first we need to store all the training data so that information can be used to extract features from them. To read all the images from a folder, I have a created a **Cell of**

size of no. of items in that particular folder, 200 in this case. Using a **For Loop, all** images were read and stored in a **Cell** declared already.

Same procedure was followed for reading and storing test data for classification.

2. Image Pre-Processing

Objective of this step is to arrange the data in such a way so that feature extraction is optimum with our computational limitations.

Resize: Images were resized to 32X32 to reduce the large data handling and computation problems. This resize operation was done using "***Nearest Neighbour Interpolation***".

Conversion into Double: Data was converted into double format for easy numerical operations in features and discriminant functions.

3. Feature Extraction:

For the given assignment, **in 1(a)** part, we were asked to choose **image pixel vectors** as features. So this choice of features was followed for all the five parts of 1(a). To achieve feature vectors (32*32X1 =1024X1) for all the images, **imreshape command** was used in a For Loop.

Additional Features was explored in "*Feature Exploration*" section. Method for extracting them is also described in that section.

4. Classifier Model Development:

After extracting the feature vectors, our job is to develop our classifier model. We are assuming that our data come from a multi Variate Normal Distribution. Thus we need to find the μ and ∑ matrices for each class. This calculation, as specified in previous sections, was done using **Maximum Likelihood Method.** These expressions were evaluated using simple matrix operation and loops in **MATLAB**.

5. Classification of Test Data:

Test data was read in the same way as training data. After reading the test data, a feature vector was created in the same way and for that particular feature vector value of **Discriminant Functions** was evaluated. According to the value of discrimination function value, the data was classified.

TO RUN Code:

I have prepared separate files for each of part of question 1(a) (named with Akshat_Bordia_11010848_Assignment1*,* is I, II, III, IV and V) and a separate file for 1(c) Bonus Assignment. One can run them by entering their names in MATLAB command prompt or by clicking on RUN Button in MATLAB Editor.

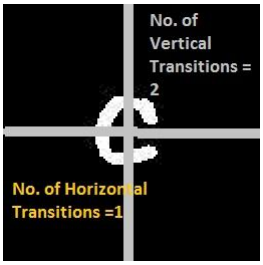
6. Additional Features Explored: Dimensionality Reduction

Transitions and Distance Features:

In this method, for each training image, we calculate following quantities for Black and White Image (converted):

1. For every row, horizontal count of transition from 0 to 1;
2. For every column, vertical count of transition from 0 to1 ;
3. For every row, the nearest pixel which is high value(1) (from right and left);
4. For every column, the nearest pixel which is high value(1)(from top and bottom);
   Demonstration of following features is given in few sample images:

(a) Transitions are shown in This sample 'C' image



(b) Nearest high pixel from Top and Bottom

(c) Nearest Pixel from Right and Left

If our image is resized to 32X32 pixels, then 1st and 2nd quantity can be expressed in a 32X1 column vector.

Similarly there will be 32X1 column vectors for each:

1. Nearest high pixel distance from right;
2. Nearest high pixel distance from left;
3. Nearest high pixel distance from top;
4. Nearest high pixel distance from bottom.

Hence overall we have 32*6X1 =192X1 column vector and 192X192 covariance matrix. **There is a large reduction in dimensionality** here in comparison to our original pixel value feature**. A regularization term 0.3*I was added to covariance matrix in order to achieve its inevitability**. This method (Separate Non Diagonal Covariance Matrix) was implemented and good results were achieved. Accuracies are listed below:

(A) For Character 'e' :

Total no. of test data = 100,
Characters classified as 'e' = 87,
Characters classified as 'c' = 12,
Characters classified as 'i' = 1,

No. of Characters Classified Correctly = 87;
No. of Characters NOT classified correctly = 12+1 =13

Accuracy for 'e'= $\frac{No.of \text{ "e" characters } which\ were\ classified\ correctly}{No.of\ Total \text{ "e" characters}} * 100,$

Accuracy = $\frac{87*100}{100} = \mathbf{87\%}$

(B) For Character 'c':

Total no. of test data = 100,
Characters classified as 'e' = 26,
Characters classified as 'c' = 73,
Characters classified as 'i' = 1,
No. of Characters Classified Correctly = 73;
No. of Characters NOT classified correctly = 26+1=27

Accuracy for 'c'= $\frac{No.of \text{ "c" characters } which\ were\ classified\ correctly}{No.of\ Total \text{ "c" characters}} * 100,$

Accuracy = $\frac{73}{100} = \mathbf{73\%}$

(C) For Character 'i' :

Total no. of test data = 100,
Characters classified as 'e' = 2
Characters classified as 'c' = 3,
Characters classified as 'i' = 95,
No. of Characters Classified Correctly = 95;
No. of Characters NOT classified correctly = 2+3=5;

Accuracy for 'i'= $\frac{No.of \text{ "i" characters } which\ were\ classified\ correctly}{No.of\ Total \text{ "i" characters}} * 100,$

Accuracy = $\frac{95*100}{100} = \mathbf{95\%}$

(D) Average Accuracy :

Average Accuracy = $\frac{Sum\ of\ Individual\ Character\ Accuracies}{3}$

Average Accuracy = $\frac{87+73+95}{3} = 85\%$

Note: To run this program, Please open the file "AkshatBordia_11010848_Assignment1C" and click on Run button in MATLAB Editor or enter the file name in MATLAB Command Prompt and press enter. The results will print true category and category given by our classifier. In the code, some

count variables are specified, which can be used to find out true classification and false classification and hence accuracy.

Please note that this code was done on MATLAB2012b and might not be compatible with other versions. In any case, I would be willing to show the instructors and teaching assistants demonstration on my PC if the code does not work properly on lab pc.

7 . Accuracy and Observations:

Regularization term 0.8*I was added to achieve the inevitability of covariance matrix. In code files, counter variables were added in loops to count how many characters are correctly classified and how many are false classified to which category. By printing the value of these counts, we can calculate accuracies.

1. **(a)(I)** the samples of a given character class are modelled by a separate covariance matrix ∑i.

   The code was tested on all 300 Images given in Test Data folder. To check the accuracy, correctly classified characters were counted in program.

Individual Character Accuracies and Average Accuracies are calculated below:

(A) For Character 'e' :

Total no. of test data = 100,
Characters classified as 'e' = 98,
Characters classified as 'c' = 1,
Characters classified as 'i' = 1,
No. of Characters Classified Correctly = 98;
No. of Characters NOT classified correctly = 2;

Accuracy for 'e'= $\frac{No.of \text{ "e" characters } which\ were\ classified\ correctly}{No.of\ Total \text{ "e" characters}} * 100,$

Accuracy = $\frac{98*100}{100} = \mathbf{98\%}$

(B) For Character 'c':

Total no. of test data = 100,
Characters classified as 'e' = 4,
Characters classified as 'c' = 92,
Characters classified as 'i' = 4,
No. of Characters Classified Correctly = 92;
No. of Characters NOT classified correctly = 4+4 =8

Accuracy for 'c'= $\frac{No.of \text{ "c" characters } which\ were\ classified\ correctly}{No.of\ Total \text{ "c" characters}} * 100,$

Accuracy = $\frac{92*100}{100}$ = **92%**

(C) For Character 'i' :

Total no. of test data = 100,
Characters classified as 'e' = 0,
Characters classified as 'c' = 0,
Characters classified as 'i' = 100,
No. of Characters Classified Correctly = 100;
No. of Characters NOT classified correctly = 0;

Accuracy for 'i'= $\frac{No.of \text{ "i" characters } which\ were\ classified\ correctly}{No.of\ Total \text{ "i" characters}} * 100,$

Accuracy = $\frac{100*100}{100}$ = **100%**

(D) Average Accuracy :

Average Accuracy = $\frac{Sum\ of\ Individual\ Character\ Accuracies}{3}$

Average Accuracy = $\frac{92+98+100}{3}$ =96.66%

1. (a)(II) The samples across all the character classes are pooled together to generate a common non diagonal covariance matrix.

1. For Character 'e' :

Total no. of test data = 100,
Characters classified as 'e' = 97,
Characters classified as 'c' = 3,
Characters classified as 'i' = 0,
No. of Characters Classified Correctly = 97;
No. of Characters NOT classified correctly = 3+0=3

Accuracy for 'e'= $\frac{No.of \text{ "e" characters } which\ were\ classified\ correctly}{No.of\ Total \text{ "e" characters}} * 100,$

Accuracy = $\frac{97*100}{100}$ = **97%**

2. For Character 'c':

Total no. of test data = 100,
Characters classified as ‘e’ = 3,
Characters classified as ‘c’ = 97,
Characters classified as ‘i’ = 0,
No. of Characters Classified Correctly = 97;
No. of Characters NOT classified correctly = 3+00=3

Accuracy for ‘c’= $\frac{No.of \text{ "c" characters } which\ were\ classified\ correctly}{No.of\ Total \text{ "c" characters}} * 100,$

Accuracy = $\frac{97*100}{100} = \mathbf{97\%}$

3. **For Character ‘i’ :**

Total no. of test data = 100,
Characters classified as ‘e’ = 0,
Characters classified as ‘c’ = 0,
Characters classified as ‘i’ = 100,
No. of Characters Classified Correctly = 100;
No. of Characters NOT classified correctly = 100+0=100;

Accuracy for ‘i’= $\frac{No.of \text{ "i" characters } which\ were\ classified\ correctly}{No.of\ Total \text{ "i" characters}} * 100,$

Accuracy = $\frac{100*100}{100} = \mathbf{100\%}$

4. **Average Accuracy :**

Average Accuracy = $\frac{Sum\ of\ Individual\ Character\ Accuracies}{3}$

Average Accuracy = $\frac{97+97+100}{3} = 98\%$

---

1. (a)(III)

The samples of a given character class are separately modelled by a diagonal covariance matrix. The diagonal entries of the matrix correspond to the variances of the individual features. The features are assumed to be independent- hence their cross variances are forced to zero.

1. **For Character ‘e’ :**

Total no. of test data = 100,
Characters classified as ‘e’ = 84,
Characters classified as ‘c’ = 10,
Characters classified as ‘i’ = 6,
No. of Characters Classified Correctly = 84;
No. of Characters NOT classified correctly = 10+6=16;

Accuracy for ‘e’= $\frac{No.of \text{ "e" characters } which\ were\ classified\ correctly}{No.of\ Total \text{ "e" characters}} * 100,$

Accuracy = $\frac{84*100}{100}$ = **84%**

2. **For Character ‘c’:**

Total no. of test data = 100,
Characters classified as ‘e’ = 8,
Characters classified as ‘c’ = 76,
Characters classified as ‘i’ = 16,
No. of Characters Classified Correctly =76,
No. of Characters NOT classified correctly = 8+16=24;

Accuracy for ‘c’= $\frac{No.of \text{ "c" characters } which\ were\ classified\ correctly}{No.of\ Total \text{ "c" characters}} * 100,$

Accuracy = $\frac{76*100}{100}$ = 76%

3. **For Character ‘i’ :**

Total no. of test data = 100,
Characters classified as ‘e’ = 0,
Characters classified as ‘c’ = 0,
Characters classified as ‘i’ = 100,
No. of Characters Classified Correctly = 100,
No. of Characters NOT classified correctly = 0,

Accuracy for ‘i’= $\frac{No.of \text{ "i" characters } which\ were\ classified\ correctly}{No.of\ Total \text{ "i" characters}} * 100,$

Accuracy = $\frac{100*100}{100}$ = **100%**

4. **Average Accuracy :**

Average Accuracy = $\frac{Sum\ of\ Individual\ Character\ Accuracies}{3}$

Average Accuracy = $\frac{84+76+100}{3}$ =86.66%

1. (a) (IV)

The samples across all the character classes are pooled to generate a common diagonal covariance matrix. The diagonal entries correspond to the variances of the individual features that are considered to be independent.

Individual Character Accuracies and Average Accuracies are calculated below:

(A) For Character 'e' :

Total no. of test data = 100,
Characters classified as 'e' = 88,
Characters classified as 'c' = 12,
Characters classified as 'i' = 0,
No. of Characters Classified Correctly = 88;
No. of Characters NOT classified correctly = 12

Accuracy for 'e'= $\frac{No.of \text{ "e" characters } which\ were\ classified\ correctly}{No.of\ Total \text{ "e" characters}} * 100,$

Accuracy = $\frac{88*100}{100}$ = **88%**

(B) For Character 'c':

Total no. of test data = 100,
Characters classified as 'e' = 09,
Characters classified as 'c' = 86,
Characters classified as 'i' = 5,
No. of Characters Classified Correctly = 86;
No. of Characters NOT classified correctly = 9+5=14

Accuracy for 'c'= $\frac{No.of \text{ "c" characters } which\ were\ classified\ correctly}{No.of\ Total \text{ "c" characters}} * 100,$

Accuracy = $\frac{86*100}{100}$ = **86%**

(C) For Character 'i' :

Total no. of test data = 100,
Characters classified as 'e' = 0,
Characters classified as 'c' = 0,
Characters classified as 'i' = 98,
No. of Characters Classified Correctly = 98;
No. of Characters NOT classified correctly = 2

Accuracy for 'i'= $\frac{No.of \text{ "i" characters } which\ were\ classified\ correctly}{No.of\ Total \text{ "i" characters}} * 100,$

Accuracy = $\frac{98*100}{100}$ = **98%**

(D) Average Accuracy :

Average Accuracy = $\frac{Sum\ of\ Individual\ Character\ Accuracies}{3}$

Average Accuracy = $\frac{88+86+98}{3}$ =90.66%

---

1. (a)(V) The covariance matrix of each class is forced to be spherical

1. For Character 'e' :

Total no. of test data = 100,
Characters classified as 'e' = 83,
Characters classified as 'c' = 8,
Characters classified as 'i' = 9,
No. of Characters Classified Correctly = 83;
No. of Characters NOT classified correctly = 8+9=17;

Accuracy for 'e'= $\frac{No.of\ \text{"e" characters}\ which\ were\ classified\ correctly}{No.of\ Total\ \text{"e" characters}}$*100,

Accuracy = $\frac{83*100}{100}$ = **83%**

2. For Character 'c':

Total no. of test data = 100,
Characters classified as 'e' = 8,
Characters classified as 'c' = 73,
Characters classified as 'i' = 19,
No. of Characters Classified Correctly =73,
No. of Characters NOT classified correctly = 8+19=27;

Accuracy for 'c'= $\frac{No.of\ \text{"c" characters}\ which\ were\ classified\ correctly}{No.of\ Total\ \text{"c" characters}}$*100,

Accuracy = $\frac{73*100}{100}$ = 73%

3. For Character 'i' :

Total no. of test data = 100,
Characters classified as 'e' = 0,
Characters classified as 'c' = 0,
Characters classified as 'i' = 100,

No. of Characters Classified Correctly = 100
No. of Characters NOT classified correctly =0

Accuracy for 'i'= $\frac{No.of\ \text{"i" characters}\ which\ were\ classified\ correctly}{No.of\ Total\ \text{"i" characters}} * 100,$

Accuracy = $\frac{100*100}{100}$ = **100%**

4. **Average Accuracy :**

Average Accuracy = $\frac{Sum\ of\ Individual\ Character\ Accuracies}{3}$

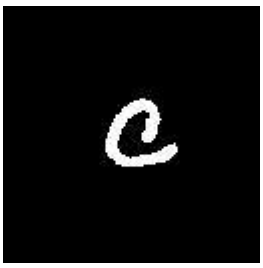
Average Accuracy = $\frac{83+73+100}{3}$ =85.33%

1(b)

(5 points) Give 4 examples of images from the test set that are misclassified by each of the classifiers designed in Task 1(a). Display both the state of nature (true label) and the predicted class for each image.

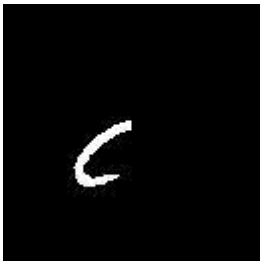
(I) **Separate Non Diagonal Covariance Matrices :**

True State "C"
Predicted "e"



True State "C"
Predicted "e"

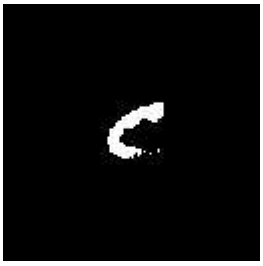
True State "C"
Predicted "i"



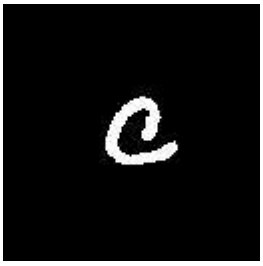
True State "C"
Predicted "i"

(II) Common Non Diagonal Matrix:

True Class: "C"
Predicted: "I"



True Class: "C"
Predicted: "e"



True Class "e"
Predicted "C"

True Class "e"
Predicted "c"

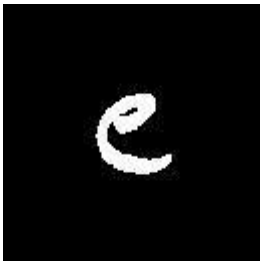
(III) Separate Diagonal Covariance Matrices :

True Class: "C", Predicted: "e" (both)

True Class "e", Predicted:"C" (both)

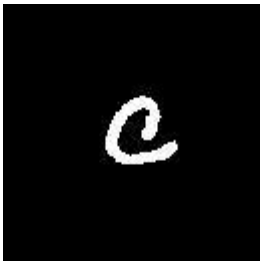
(IV) Common Diagonal Matrix:

True Class "e"
Predicted: "c"

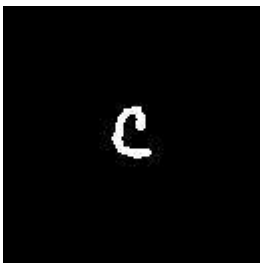


True Class: "e"
Predicated "c"

True Class: "C"
Predicted "e"

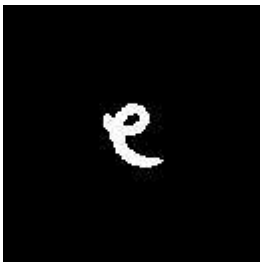


True Class :"C"
Predicted "l"



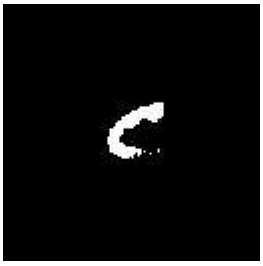
(V) Spherical Covariance Matrix Case:

True Class: "e"
Predicted: "c"

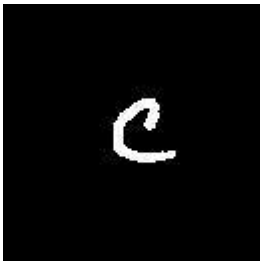


True Class: "e"
Predicted: "i"

True Class: "C"
Predicted: "i"



True Class "C"
Predicted : "e"



Bibliography:

1. Duda, R. O., et al. (2012). Pattern classification, John Wiley & Sons.
2. John A. Gubner. Probability and Random Processes for Electrical and Computer Engineers
3. Vamvakas, Georgios, et al. "An efficient feature extraction and dimensionality reduction scheme for isolated Greek handwritten character recognition." *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*. Vol. 2. IEEE, 2007.