ECE8560

Pattern Recognition

Takehome #1

Design and Implementation of Bayesian Classifier

Shreyas Prabhakar

sprabha

1. Overview

In this take-home quiz we were asked to design and test a Bayesian, minimum error classifier for a given H and S_T.

The problem is a c=3 class, d=4 dimensional problems. The training data and testing data were provided to us. Training data of 15000 samples were provided to us in a file name train_sp2017_v19. The first 5000 samples belong to 1st class *w1 (H1),* the next 5000 samples belong to the 2nd class *w2 (H2)* and the remaining 5000 samples belong to the 3rd class *w3 (H3).*

I have used MATLAB to implement and the Bayesian, minimum error classifier.

2. Bayesian Classifier

The main task for designing a classifier is developing decision or classification strategies. We use Bayes theorem to convert a priori estimate of the probability $P(w_i)$ to a posteriori probability $P(w_i|x)$, or measurement conditioned, probability of a state of nature. The Bayes theorem is given as

$$P(w_i|x) = \frac{p(x/w_i)P(w_i)}{p(x)}$$

$$\text{where,} \quad p(x) = \sum p(x/w_i)P(w_i)$$

1. Organising Training Data

My first task in designing the Bayesian classifier was to divide this training data into training set and test set. The test set will be used later to test our classifier system and to find the error. I decided to divide the first 90% of the sample in class as **training set** (i.e., **4500 samples per class**. Total of 13500 samples). And the remaining 10% of the data for **testing** (i.e., **500 samples per class**. Total of 1500 samples).

2. Maximum Likelihood Estimation

The maximum likelihood function is used to calculate the initial mean and covariance matrices of all the three classes

The mean and covariance estimates are given by

$$\mu_i = \frac{1}{n}\sum_{k=1}^{n} x_k$$

$$\sum = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - \mu)(x_k - \mu)^T$$

3. Bayesian Parameter Estimation

For the Bayesian parameter estimation, we assume that the covariance matrix is known. We use the covariance matrix Σ estimated using the Maximum Likelihood Estimation for this.

The mean and covariance of the multivariate Gaussian case is given by

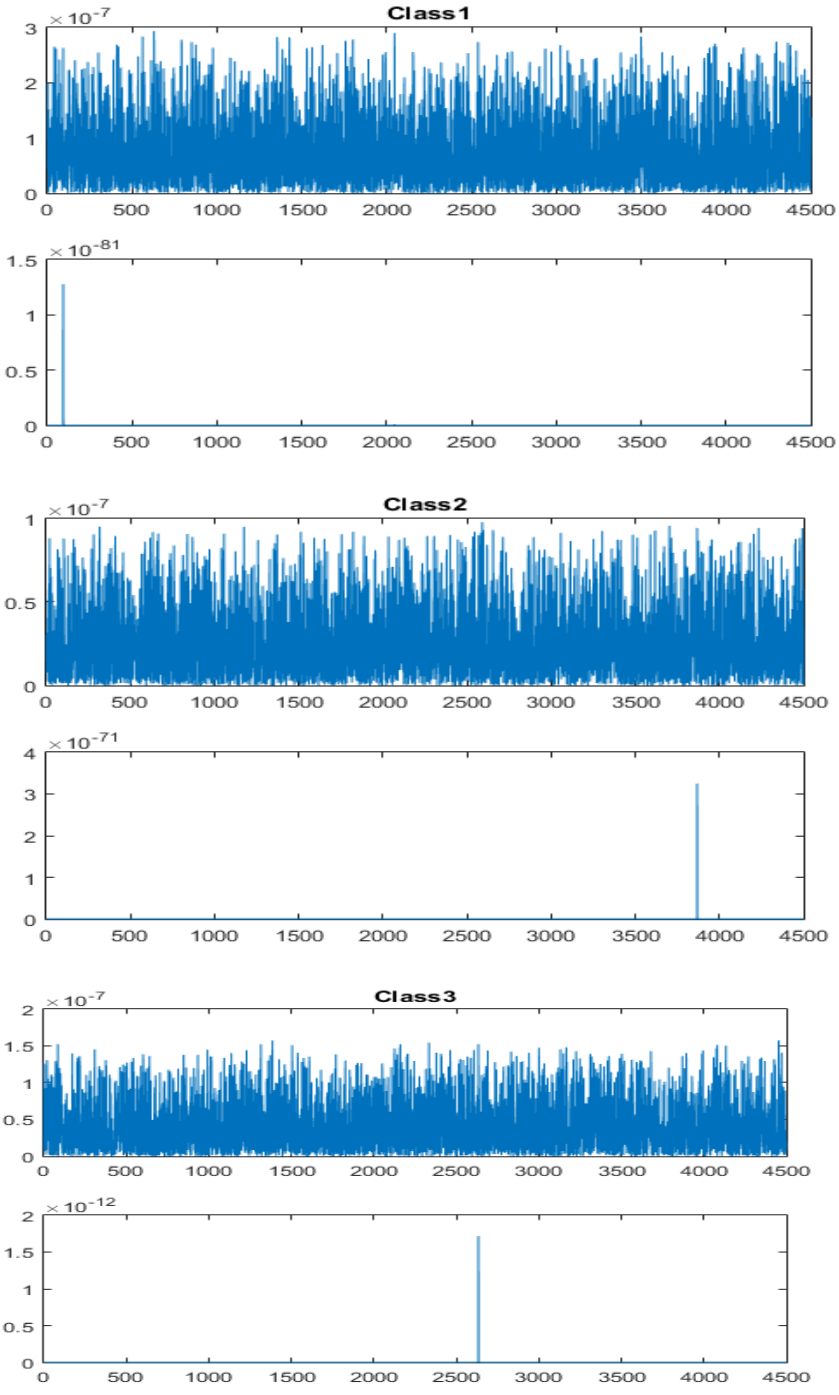
$$\mu_n = \Sigma_0(\Sigma_0 + \frac{1}{n}\Sigma)^{-1}\frac{1}{n}\Sigma_{k=1}^{n} x_k \, \Sigma_0(\Sigma_0 + \frac{1}{n}\Sigma)^{-1}\mu_0$$

$$\Sigma_n = (\Sigma_0(\Sigma_0 + \frac{1}{n}\Sigma)^{-1}\frac{1}{n}\Sigma)$$

where, $\Sigma_0 = I$ (assumed)

$\mu = 0$ (assumed)

The plots below show that using Hi, we can refine $p_{\theta i}(\theta_i)$ to be an impulse function



4. Discriminant Function

The discriminant function for the i^{th} class is defined as

$$g_i(x) = P(w_i|x)$$

The discriminant function $g_i(x)$ gives the decision rules for classification. The decision boundaries are calculated by using the general (Gaussian case) result for the discriminant function $g_i(x)$. Given by

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^T \sum_i^{-1} (x-\mu_i) - \left(\frac{d}{2}\right)\log(2\pi) - \frac{1}{2}\log|\sum_i|$$

The first term in the expression represents the square of the Mahalanobis distance from x to μ_i. The second term is a class independent term and can be neglected. The equation then reduces to

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^T \sum_i^{-1} (x-\mu_i) - \frac{1}{2}\log|\sum_i|$$

For a given x, we calculate the discriminant function $g_i(x)$ for all the classes i. We then make a decision saying that the vector x falls in the class which has the maximum $g_i(x)$.

3. Error Estimation.

Once the training of the Bayesian classifier was done with a total of 13500 samples (4500 samples from each class), it was asked to classify the testing set of 1500 samples (500 samples from each class) whose class was known prior to the classification.
The error was found to be **8.93%.**
The probability of error was calculated using
P(E) = number of samples wrongly classified / number of samples

APPENDIX A:

MATLAB CODE:

```
clear all

fp = fopen('train_sp2017_v19');
A = textscan(fp,'%f%f%f%f');
[~] = fclose(fp);

dim = size(A,2);
totalSize = size(A{1,1},1);
C = 3;
sizeC = totalSize/3;
N = 0.9 * sizeC;
for i = 1:dim
H(:,i) = A{1,i}(1:end, :);
end
clear A fp

for i = 1:C
    starting = ((i-1)*sizeC)+1;
    ending = ((i-1)*sizeC)+N;
    w{1,i} = H(starting:ending,:);
end
clear starting ending

%Maximum Likelihood Estimation
for i = 1:C
MLEm{1,i} = sum(w{1,i})/N;
MLEc{1,i} = cov(w{1,i});
end

sig0 = eye(dim);
mu0 = zeros(1,dim)';

% Bayesian Parameter Estimation
for i = 1:C
   b = (1/N) * MLEc{1,i};
   a = inv(sig0 + b);
   sign{1,i} = sig0 * a * b;
   mu{1,i} = ((sig0 * a * MLEm{1,i}') + (b * a * mu0))';
end
clear a b

%Classifying the test data
ftest = fopen('train_sp2017_v19');
A = textscan(ftest,'%f%f%f%f');
[~] = fclose(ftest);

for i = 1:dim
xi(:,i) = A{1,i}(1:end, :);
end

clear ftest A

%Discriminant Function g(i)
for i = 1 : size(xi, 1)
    for j = 1 : C
```

```
        mahal_dist = ((xi(i,:)-mu{1,j}) * inv(sign{1,j}) * (xi(i,:)-
mu{1,j})');
        logc = log(det(sign{1,j}));
        g(i,j) = -(1/2) * (mahal_dist + logc);
    end

    if ((g(i, 1) >= g(i, 2)) && (g(i, 1) >= g(i, 3)))
        predict(i) = 1;
    elseif ((g(i, 2) >= g(i, 1)) && (g(i, 2) >= g(i, 3)))
        predict(i) = 2;
    else
        predict(i) = 3;
    end
end

fx = fopen('x.dat', 'w');
for i = 1 : length(predict)
        fprintf(fx, '%d\n', predict(i));
end
[~] = fclose(fx);
```