Contents

Vincent Purcell - HW 1 - ECE487

```
clear; clc; close all;
```

Problem 2.7

```
% Initializing the variables used for functions
m = [[1;1],[4;4],[8;1]];
S(:,:,1) = 2.*[1,0;0,1];
S(:,:,2) = 2.*[1,0;0,1];
S(:,:,3) = 2.*[1,0;0,1];
P1 = [0.333;0.333;0.334];
P2 = [0.8;0.1;0.1];
N = 1000;

% Generated Gaussian classes
[X,y] = generate_gauss_classes(m,S,P1,N);
[X_prime,y_prime] = generate_gauss_classes(m,S,P2,N);

% Classify the Equiprobable and Unevenly Distributed Datasets using
% Euclidean and Bayesian Classification functions
z_bc = bayes_classifier(m,S,P1,X');
z_ec = euclidean_classifier(m,X');
z_prime_bc = bayes_classifier(m,S,P2,X_prime');
z_prime_ec = euclidean_classifier(m,X_prime');
```

Compute Error for Bayesian and Euclidean Classifications of X

```
z_bc_error = compute_error(z_bc,y);
z_ec_error = compute_error(z_ec,y);
```

Compute Error for Bayesian and Euclidean Classifications of X Prime

```
z_prime_bc_error = compute_error(z_prime_bc,y_prime);
z_prime_ec_error = compute_error(z_prime_ec,y_prime);
```

Conclusion for Problem 2.7

Based on the Errors calculated above and printed out below I could make the following conclusions. For the equiprobable dataset the euclidean and bayesian classifications produce extremely similar results and thus the same error. For the dataset with 0.8,0.1,0.1 class distribution the bayesian classification produces much better results than the euclidean classification.

```
fprintf('Bayesian Classification error for Equiprobable Dataset: %.3f\n', z_bc_error)
fprintf('Euclidean Classification error for Equiprobable Dataset: %.3f\n', z_ec_error)
fprintf('Bayesian Classification error for Unevenly Distributed Dataset: %.3f\n', z_prime_bc_error)
fprintf('Euclidean Classification error for Unevenly Distributed Dataset: %.3f\n', z_prime_ec_error)
```

```
Bayesian Classification error for Equiprobable Dataset: 0.075
Euclidean Classification error for Equiprobable Dataset: 0.075
Bayesian Classification error for Unevenly Distributed Dataset: 0.036
Euclidean Classification error for Unevenly Distributed Dataset: 0.070
```

Problem 2.8

```
% Initializing the variables used for functions
m = [[1;1],[8;6],[13;1]];
S(:,:,1) = 6.*[1,0;0,1];
S(:,:,2) = 6.*[1,0;0,1];
S(:,:,3) = 6.*[1,0;0,1];
P1 = [0.333;0.333;0.334];
N = 1000;

% Generated Gaussian classes
[X,y] = generate_gauss_classes(m,S,P1,N);
[Z,y_z] = generate_gauss_classes(m,S,P1,N);

% Classify class using K Nearest Neighbors with k=1 and calcualte
% classification error, X was used as the testing set and Z was used as the
% training set
z_1 = k_nn_classifier(Z',y_z,1,X');
z_1_error = compute_error(z_1,y);

% Classify class using K Nearest Neighbors with k=11 and calculate
% classification error
z_11 = k_nn_classifier(Z',y_z,11,X');
z_11_error = compute_error(z_11,y);
```

Conclusion for Problem 2.8

Based on the error values found above. One could conclude that when using the K Nearest Neighbors Algorithm if you increase k the classification error would go down. Below the is the classification error of X for k equal to 1 and k equal to 11.

```
fprintf('K Nearest Neighbor Classification error of X with k=1: %.3f\n', z_1_error)
fprintf('K Nearest Neighbor Classification error of X with k=11: %.3f\n', z_11_error)
```

```
K Nearest Neighbor Classification error of X with k=1: 0.107
K Nearest Neighbor Classification error of X with k=11: 0.089
```

Functions Received From Textbook

The following functions were received from the Textbook
Pattern Recognition - Theodoridis, Koutroumbas

Generate Gaussian Classes Function

Received from page 80 of the Text

```
function [X,y]=generate_gauss_classes(m,S,P,N)
    [l,c]=size(m);
    X=[];
    y=[];
    for j=1:c
    % Generating the [p(j)*N)] vectors from each distribution
        t=mvnrnd(m(:,j),S(:,:,j),fix(P(j)*N));
        % The total number of points may be slightly less than N
        % due to the fix operator
        X=[X;t];
        y=[y ones(1,fix(P(j)*N))*j];
    end
end
```

Bayes Classifier Function

Received from page 81 of the Text

```
function z=bayes_classifier(m,S,P,X)
    [l,c]=size(m); % l=dimensionality, c=no. of classes
    [l,N]=size(X); % N=no. of vectors
    for i=1:N
        for j=1:c
            t(j)=P(j)*comp_gauss_dens_val(m(:,j),...
                S(:,:,j),X(:,i));
        end
        % Determining the maximum quantity Pi*p(x|wi)
        [num,z(i)]=max(t);
    end
end
```

Gaussian Function Evaluation Function

Received from page 79 of the Text

```
function z=comp_gauss_dens_val(m,S,x)
    [l,q]=size(m); % l=dimensionality
    z=(1/((2*pi)^ (l/2)*det(S)^ 0.5) )...
        *exp(-0.5*(x-m)'*inv(S)*(x-m));
end
```

Euclidean Classifier Function

Received from page 82 of the Text

```
function z=euclidean_classifier(m,X)
    [l,c]=size(m); % l=dimensionality, c=no. of classes
    [l,N]=size(X); % N=no. of vectors
    for i=1:N
        for j=1:c
            t(j)=sqrt((X(:,i)-m(:,j))'*(X(:,i)-m(:,j)));
        end
        % Determining the maximum quantity Pi*p(x|wi)
        [num,z(i)]=min(t);
    end
end
```

Compute Classification Error Function

Received from page 83 of the Text

```
function clas_error=compute_error(y,y_est)
    [q,N]=size(y); % N= no. of vectors
    c=max(y); % Determining the number of classes
    clas_error=0; % Counting the misclassified vectors
    for i=1:N
        if(y(i)~=y_est(i))
            clas_error=clas_error+1;
        end
    end
    % Computing the classification error
    clas_error=clas_error/N;
end
```

K Nearest Neighbor Classification Function

Received from page 83 of the Text

```
function z=k_nn_classifier(Z,v,k,X)
    [l,N1]=size(Z);
    [l,N]=size(X);
    c=max(v); % The number of classes
    % Computation of the (squared) Euclidean distance
    % of a point from each reference vector
    for i=1:N
        dist=sum((X(:,i)*ones(1,N1)-Z).^ 2);
        %Sorting the above distances in ascending order
        [sorted,nearest]=sort(dist);
        % Counting the class occurrences among the k-closest
        % reference vectors Z(:,i)
        refe=zeros(1,c); %Counting the reference vectors per class
        for q=1:k
            class=v(nearest(q));
            refe(class)=refe(class)+1;
        end
        [val,z(i)]=max(refe);
    end
end
```

The Sufficient Statistic

In order to understand this concept of sufficient statistics I first watched a video on YouTube named "An Introduction to the concept of a sufficient statistic." This video detailed a Bernoulli Distribution of parameter theta that explains a coin flip scenario. The process to find a sufficient statistic took the Bernoulli distribution function which is a function of the parameter theta and of the data points x_i. Once this Product Summation of the likelihood is found from the Probability Density Function, it can be written out and then you could find a statistic within the equation that is solely affected by the parameter theta. This statistic thus provides enough information about the distribution, and the rest of the equation provides no additional information. I then went on itl.nist.gov and found their "Gallery of Distributions" in order to find the probability density functions for the various distribution types.

Gaussian

First, I looked at the probability density function of a Gaussian Distribution:

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\theta)^2 / 2\sigma^2}$$

Theta in the density equation above if the parameter that effects the sufficient statistic. If the density equation above is written as a product summation where *x* is the feature value, we could then find the sufficient statistic.

$$F(x) = \prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} e^{-(x_i-\theta)^2 / 2\sigma^2}$$

When this product summation is written out the value in the exponential seen above $-(x_i-\vartheta)^2$ can be written as the following after using factorization theorems:

$$(\theta - \bar{x})^2$$

Then we can see that the parameter is solely affected by the mean of the features and thus the mean of the features is the sufficient statistic

Poisson

The Poisson distribution requires much simpler math then the Gaussian distribution. Again, the probability density function is found and then written as a product summation to find the likelihood:

$$P(x;\theta) = \prod_{i=1}^{N} \frac{e^{-\theta}\theta^{x_i}}{x_i!}$$

When this product summation is written out and evaluated the only value within the equation that effects the parameter is ϑ^{xi} and with basic exponential algebra the following sufficient statistic is found:

$$\theta^{x_1} + \theta^{x_2} + \cdots + \theta^{x_N} = \theta^{x_1+x_2+\cdots+x_N}$$

And thus the sufficient statistic can be found to be the sum of the features or with respect to the mean it will be the mean multiplied by the amount of features:

$$sufficient\ statistic = \sum_{i=1}^{N} x_i = N * \bar{x}$$

Exponential

The Exponential distribution requires a similar process to the Poisson distribution. Again, the probability density function was written as a product summation to find the likelihood:

$$F(x;\theta) = \prod_{i=1}^{N} \frac{1}{\theta} e^{\frac{\mu-1}{\theta}x_i}$$

With this product summation again, we must look at the exponential to find the sufficient statistic. The term in the equation above that provides sufficient information with respect to the parameter theta is the following:

$$e^{\frac{\mu-1}{\theta}x_1} + e^{\frac{\mu-1}{\theta}x_2} + \cdots + e^{\frac{\mu-1}{\theta}x_N} = e^{\frac{\mu-1}{\theta}x_1 + \frac{\mu-1}{\theta}x_2 + \cdots + \frac{\mu-1}{\theta}x_N}$$

And thus the sufficient statistic again will be:

$$sufficient\ statistic = \sum_{i=1}^{N} x_i = N * \bar{x}$$

References

- Pattern Recognition - AP (2008) - Theodoridis S., Koutroumbas K.
- 1.3.6.6 Gallery of Distributions from itl.nist.gov
    - https://www.itl.nist.gov/div898/handbook/eda/section3/eda366.htm
- An Introduction to the concept of a sufficient statistic – YouTube Video
    - https://www.youtube.com/watch?v=5j4E2FRR384
- Sufficient Statistics and Exponential Family Lecture Notes
    - http://people.missouristate.edu/songfengzheng/Teaching/MTH541/Lecture%20notes/Sufficient.pdf