Lab 2

Albert Segarra Roca (S3255050), Carlos Humberto Paz Rodríguez (S3040577)

Group 16

Pattern Recognition

FMNS • RUG

November 23, 2017

1 Assignment 1

1.1 Exercise 1

Since each row in the vector is a measurement, we capture them as the rows in the matrix, and we calculate the average of each of the three features:

Command: `mean(matr)`

Result:

5.8000 5.0000 6.2000

And with the following command we calculate the variance-covariance matrix.

Command: `cov(matr)`

Result:

3.2000 0.2500 -0.4500
0.2500 2.5000 -3.7500
-0.4500 -3.7500 5.7000

The main diagonal gives us the detail of how spread the data is on each feature, so the third feature varies more than the other two. We can also conclude that the relation between the third and the other two features is inverse, which means that if one them increases, the third feature decreases and viceversa.

1.2 Exercise 2

Considering the following matrix of values (Table 1), where each row is a feature and each column is a measurement, we want to model the information by a normal distribution.

4	6	8	7	4
5	3	7	4	6
6	9	3	8	5

Table 1: Matrix of measurements for each feature

If we were talking of one variable, we could have plotted a simple PDF function line on an x-y, and if it was 2 variables scenario, we could have plotted a 3D surface normal distribution function, but on this scenario with 3 features, we need to use an ellipsoid object to be able to model the behaviour of the information and its correlations, resulting in Figure 1. Where the center of object is the mean of the three features and we need to plot what the behaviour would be at a certain standard deviation of distance from the center, to

visualize how the values behave in all directions. We also need to make use of eigen vectors and eigen values to be able to rotate the object in the 3D space and model.

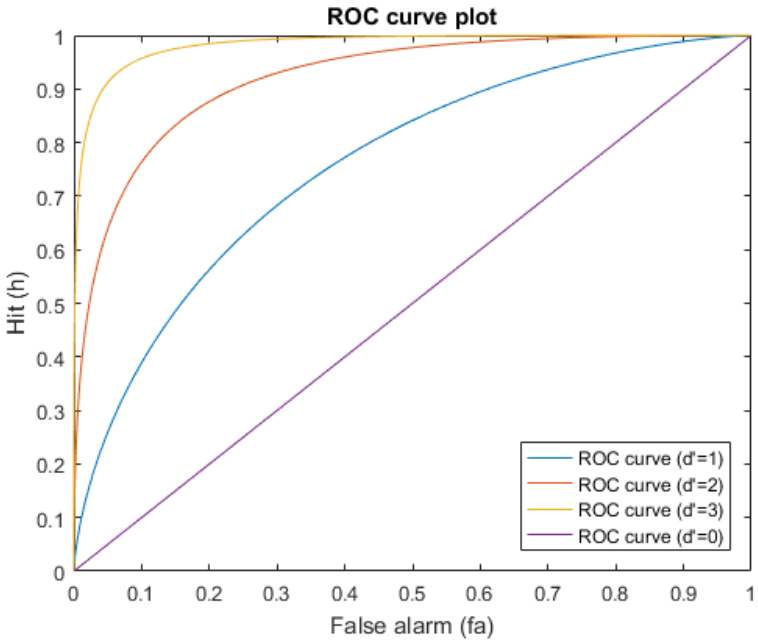

Figure 1: 3 variables normal probability density function with 1 standard deviation, and mean vector (5.8 5 6.2)

We can calculate the Probability Density Function of any point, this value means the probability that any other measurement results in this point, the sum of all the probabilities must be equal to 1. We get Table 2.

Command: `F1 = mvnpdf(x1,mn,cvr);`

Table 2: Probability Density Value of 3 variables

feature 1	feature 2	feature 3	P.D.F.
5	5	6	0.0543
3	5	7	6.1287e-04
4	6.5	1	7.0300e-29

2 Assignment 2

2.1 Exercise 1

We can represent the vectors in a matrix M, where X and Y are the column vectors representing the different measurements of each the features/variables X and Y respectively (the two features being considered):

$$M = \begin{array}{c} \begin{matrix} X & Y \end{matrix} \\ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \end{array}$$

Now we compute the mean vector of M, $\bar{V}$:

$$\bar{V} = \begin{array}{c} \begin{matrix} \bar{X} & \bar{Y} \end{matrix} \\ \begin{pmatrix} \frac{a+c}{2} & \frac{b+d}{2} \end{pmatrix} \end{array}$$

Now, we can apply the covariance formula (1), where N = number of measurement$s$ = rows of M = 2 in our case.

$$cov_{X,Y} = \frac{\sum_{i=1}^{N}(X_i - \bar{X})(Y_i - \bar{Y})}{N-1} \tag{1}$$

We can compute the covariance matrix as:

$$S = \begin{bmatrix} cov_{X,X} & cov_{X,Y} \\ cov_{Y,X} & cov_{Y,Y} \end{bmatrix}$$

Applying the formula, we obtain:

$$S = \begin{bmatrix} \frac{(a-\frac{a+c}{2})^2+(c-\frac{a+c}{2})^2}{2-1} & \frac{(a-\frac{a+c}{2})(b-\frac{b+d}{2})+(c-\frac{a+c}{2})(d-\frac{b+d}{2})}{2-1} \\ \frac{(b-\frac{b+d}{2})(a-\frac{a+c}{2})+(d-\frac{b+d}{2})(c-\frac{a+c}{2})}{2-1} & \frac{(b-\frac{b+d}{2})^2+(d-\frac{b+d}{2})^2}{2-1} \end{bmatrix}$$

Simplifying:

$$S = \begin{bmatrix} \frac{(a-c)^2}{2} & \frac{(a-c)(b-d)}{2} \\ \frac{(a-c)(b-d)}{2} & \frac{(b-d)^2}{2} \end{bmatrix}$$

We can see that the covariance matrix S is symmetric, due to the product distributivity.

2.2 Exercise 2

In order to compute the new matrix, we can substitute the new values in the matrix obtained in the previous exercise:

$$S' = \begin{bmatrix} \frac{(a+k-c-k)^2}{2} & \frac{(a+k-c-k)(b+k-d-k)}{2} \\ \frac{(a+k-c-k)(b+k-d-k)}{2} & \frac{(b+k-d-k)^2}{2} \end{bmatrix}$$

Simplifying:

$$S' = \begin{bmatrix} \frac{(a-c)^2}{2} & \frac{(a-c)(b-d)}{2} \\ \frac{(a-c)(b-d)}{2} & \frac{(b-d)^2}{2} \end{bmatrix} = S$$

We see that all the constants k cancel, and we get the same covariance matrix S, so $S = S'$. This means that the covariance matrix is unaffected when we add the same constant to all the measurement values. That is because the covariance expresses the linear relationship between two variables, and adding a constant doesn't change the linear relationship, it does only displace the values.

2.3 Exercise 3

Again, we do the same as in the previous exercise:

$$S'' = \begin{bmatrix} \frac{(a*k-c*k)^2}{2} & \frac{(a*k-c*k)(b*k-d*k)}{2} \\ \frac{(a*k-c*k)(b*k-d*k)}{2} & \frac{(b*k-d*k)^2}{2} \end{bmatrix}$$

Simplifying:

$$S'' = \begin{bmatrix} k^2\frac{(a-c)^2}{2} & k^2\frac{(a*k-c*k)(b*k-d*k)}{2} \\ k^2\frac{(a*k-c*k)(b*k-d*k)}{2} & k^2\frac{(b*k-d*k)^2}{2} \end{bmatrix} = k^2 \begin{bmatrix} \frac{(a-c)^2}{2} & \frac{(a-c)(b-d)}{2} \\ \frac{(a-c)(b-d)}{2} & \frac{(b-d)^2}{2} \end{bmatrix} = k^2 S$$

In this case we see that when we scale all the measurement values byu the same constant, the covariance matrix gets scaled by the square of that constant. This is again because the covariance expresses the linear relationship between the variables, and so the following property holds:

$\text{cov}(aX, bY) = ab\ \text{cov}(X, Y)$

In our case $a = b = k$, so $a * b = k^2$

3 Assignment 3

3.1 Exercise 1

Using a mean value of [3 4] and a covariance matrix of [1 0;0 2] we plot (Figure 2) a mesh figure to view the probability distribution of the normal Gaussian distribution.

Command: `mesh(X,Y,F)`

Figure 2: Normal probability distribution using mesh plot, of two variables

3.2 Exercise 2

In order to calculate the Mahalanobis distance of certain values, we first need to generate a sample of points that belong to the normal distribution curve, and based on this sample we use the Matlab function to calculate the Mahalanobis distance, wich uses this sample data to caulculate the center of mass position, and starting from there, it calculates the distance of each value, therefore the more sample points we use, the result will be more precise, on our test we used 100,000 sample points from the P.D.F. surface. We obtain Figure 3.

Command: `d1 = mahal(Y,X)`

Figure 3: Mahalanobis distance from a multivariate normal distribution function, using 100,000 sample data

4 Assignment 4

4.1 Exercise 1

The simulation is performed using the C++ program attached in the appendix (Section 7.5.1).

Note that this program performs the simulation with a slightly different algorithm. Instead of each turn making every player toss a coin, every player plays all its turns consecutively. This simplifies the implementation, because now we don't need to save which was the ending position of the last round for every player, and it doesn't change the result, because of the independence of the random tosses.

The result of the C++ program is saved on a file, which is then loaded in matlab to plot the result (Section 7.5.2).

Figure 4: Plot of the number of players ending in a specific endpoint

As we can see in Figure 4 this distribution resembles a **binomial probability distribution**.

That is because what the simulation does is perform **n=100** repeated trials (coin tosses), where each trial can result in just two possible outcomes (advance/don't move). Also, the probability of advancing in each trial is constant **p=0.5**, and each trial is independent. This are the conditions that every binomial experiment fulfills, and if we sum up all the successes over the n trials, we get a binomial random variable which follows a binomial distribution, which is our case.

The **mean** and **variance** of a binomial distribution can be calculated as:

$mean = n * p$

$variance = n * p * (1 - p)$

In our case:

$mean = 100 * 0.5 = 50$

$variance = 100 * 0.5 * (1 - 0.5) = 25$

Which makes sense, as we can see in the Figure 4, as the distribution is centered in the endpoint 50 (mean).

5 Assignment 5

5.1 Exercise 1

Since we are analysing normal distributions we can start with the general discriminant function for normal densities.

$$g_i(x) = lnp(x|\omega_i) + lnP(\omega_i) \tag{2}$$

Using the bayesian decision theory, given that we are dealing with 2 dimension case, we can use the following formula

$$g_i(x) = -\frac{1}{2}(x - \mu_i)^t \Sigma_i^{-1}(x - \mu_i) - \frac{1}{2} ln\,|\Sigma_i| + lnP(\omega_i) \tag{3}$$

Now considering the initial mean and covariance matrix, we can generate the discriminant functions.

Case i:

$\Sigma_i = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$

$\mu_i = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$

$\Sigma_i^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{4} \end{pmatrix}$

$|\Sigma_i| = 4$

The discriminant function is:

$g_i(x) = -\frac{1}{2}(x_1 - 3, x_2 - 5) \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{4} \end{pmatrix} \begin{pmatrix} x_1 - 3 \\ x_2 - 5 \end{pmatrix} - \frac{1}{2} ln(4) + ln(0.3)$

$g_i(x) = -\frac{1}{2}(x_1 - 3, x_2 - 5) \begin{bmatrix} x_1 - 3 \\ \frac{x_2 - 5}{4} \end{bmatrix} + ln(0.15)$

$g_i(x) = -\frac{(x_1 - 3)^2}{2} - \frac{(x_2 - 5)^2}{8} + ln(0.15)$

Case j:

$$\Sigma_j = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mu_j = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$\Sigma_j^{-1} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & 1 \end{pmatrix}$$

$$|\Sigma_j| = 2$$

The discriminant function is:

$$g_j(x) = -\tfrac{1}{2}(x_1 - 2, x_2 - 1) \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 - 2 \\ x_2 - 1 \end{pmatrix} - \tfrac{1}{2} ln(2) + ln(0.7)$$

$$g_j(x) = -\tfrac{1}{2}(x_1 - 2, x_2 - 1) \begin{bmatrix} \frac{x_1 - 2}{2} \\ x_2 - 1 \end{bmatrix} - ln(\sqrt{2}) + ln(0.7)$$

$$g_j(x) = -\tfrac{(x_1-2)^2}{4} - \tfrac{(x_2-1)^2}{2} - ln(\sqrt{2}) + ln(0.7)$$

5.2 Exercise 2

After we have the two discriminant functions, we use Matlab to solve them and we get the results (4,0) and (4,0).

And if we plot the final equation, we obtain Figure 5.

Figure 5: Plot of the decision boundary between the classes ω_i and ω_j

6 Assignment 6

6.1 Exercise 1

We will use a naive Bayes classifier, which can be represented by the following formula, which assigns a class label $\hat{\mathbf{y}} = \mathbf{C_k}$ for some $\mathbf{k}$ as follows:

$$\hat{\mathbf{y}} = \underset{\mathbf{k} \in \{1, \ldots, K\}}{\operatorname{argmax}} \; \mathbf{p}(\mathbf{C_k}) \prod_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{p}(\mathbf{x_i} | \mathbf{C_k})$$

Where C represents the sequence of classes and x represents the sequence of features (keywords that appear in the email texts in our case). So in our problem, $K = 2$, $C_0 = spam$, $C_1 = \neg spam$ and $n = |x|$ is the number of keywords in the text.

Note that the evidence doesn't appear in the product. This is because we are not interested in the actual

value of the posterior probability, but just in a value that we can use to compare between different classes, and because the evidence denominator is the same in all of them, we don't need it for the comparison.

As of the statement, $p(C_0) = 0.9$, $p(C_1) = 0.1$

Note that the probabilities $p(x_i|C_i)$ are obtained from the table 1 in the statement document.

(a) **Sentence:** We offer our dear `customers` a wide selection of classy `watches`

$x = \{customers, watches\}$

$n = |x| = 2$

Now we calculate the value of the product for every class.

$\mathbf{C_0}$ = **spam**:

$value_{C_0} = p(C_0)\ p(x_0|C_0)\ p(x_1|C_0) = 0.9 * 0.005 * 0.0003 =$ **0.00000135**

$\mathbf{C_1}$ = ¬**spam**:

$value_{C_1} = p(C_1)\ p(x_0|C_1)\ p(x_1|C_1) = 0.1 * 0.0001 * 0.000004 =$ **0.00000000004**

As we can see, $value_{C_0}$ is the maximum, so with the classifier we conclude that $\hat{y} = C_0 =$ **spam is the class of this email text**.

(b) **Sentence:** Did you have `fun` on `vacation`? I sure did!

$x = \{fun, vacation\}$

$n = |x| = 2$

Again, we calculate the value of the product for every class.

$\mathbf{C_0}$ = **spam**:

$value_{C_0} = p(C_0)\ p(x_0|C_0)\ p(x_1|C_0) = 0.9 * 0.00015 * 0.00025 =$ **0.00000003375**

$\mathbf{C_1}$ = ¬**spam**:

$value_{C_1} = p(C_1)\ p(x_0|C_1)\ p(x_1|C_1) = 0.1 * 0.0007 * 0.00014 =$ **0.0000000098**

Again in this case, $value_{C_0}$ is the maximum, so with the classifier we conclude that $\hat{y} = C_0 =$ **spam is the class of this email text**.

7 Appendix

7.1 Script for assignment 1, this script models and plots a multivariate normal P.D.F., and calculates the P.D.F. for particular values

```
function assignment1()
    matr=[4 5 6;6 3 9;8 7 3;7 4 8;4 6 5];
    mn=mean(matr);
    cvr=cov(matr);
    [U,L] = eig(cvr);
    % For N standard deviations spread of data, the radii of the eliipsoid will
    % be given by N*SQRT(eigenvalues).
    N = 1;
    radii = N*sqrt(diag(L));
```

```
10
11      % generate data for "unrotated" ellipsoid
12      [xc,yc,zc] = ellipsoid(0,0,0,radii(1),radii(2),radii(3));
13
14      % rotate data with orientation matrix U and center M
15      a = kron(U(:,1),xc);
16      b = kron(U(:,2),yc);
17      c = kron(U(:,3),zc);
18      data = a+b+c;
19      n = size(data,2);
20      x = data(1:n,:)+mn(1);
21      y = data(n+1:2*n,:)+mn(2);
22      z = data(2*n+1:end,:)+mn(3);
23
24      % now plot the rotated ellipse
25      sc = surf(x,y,z);
26      shading interp
27      title('Multivariate probability density distribution')
28      axis equal
29      xlabel('Feature 1');
30      ylabel('Feature 2');
31      zlabel('Feature 3');
32      alpha(0.5)
33
34      hold on
35
36      scatter3(mn(1),mn(2),mn(3));
37      hold off
38      %=========================================================
39
40      %Compute the multivariance probability density function
41      x1=[5 5 6];
42      x2=[3 5 7];
43      x3=[4 6.5 1];
44      F1 = mvnpdf(x1,mn,cvr);
45      F2 = mvnpdf(x2,mn,cvr);
46      F3 = mvnpdf(x3,mn,cvr);
47      disp(F1)
48      disp(F2)
49      disp(F3)
50
51  end
```

7.2 Script for assignment 3, exercise 1, given an initial 2-dimension mean vector and covariance matrix, we plot the normal distribution function using the mesh function

```
1   function assignment3ex1()
2       mu = [3 4];
3       Sigma = [1 0;0 2];
4       x1 = -10:.2:10;
5       x2 = -10:.2:10;
6       [X,Y] = meshgrid(x1,x2);
7       F = mvnpdf([X(:) Y(:)],mu,Sigma);
8       F = reshape(F,length(x2),length(x1));
9
10      mesh(X,Y,F)
11      xlabel('X')
12      ylabel('Y')
```

```
    title('Probability Density Function of 2 variables using mesh plot')
    zlabel('Probability Density')
end
```

7.3 Script for assignment 3, exercise 2, given an initial 2-dimension mean vector and covariance matrix, calculate the center of mass and the Mahalanobis distance of some sample points to this center, finally we plot the results.

```
function assignment3ex2()
%Given a mean and a covariance matrix, we generate a set of random numbers
%that fall within the normal distribution curve
mu = [3 4];
sigma = [1 0;0 2];
R = chol(sigma);
X = repmat(mu,100000,1) + randn(100000,2)*R;

%With the new generated values, we calculate the Mahalanobis distance
Y = [10 10;0 0;3 4;6 8];
d1 = mahal(Y,X)
scatter(X(:,1),X(:,2))
hold on
scatter(Y(:,1),Y(:,2),100,d1,'*','LineWidth',2)
xlabel('X')
ylabel('Y')
title('Mahalanobis distance plot')
hb = colorbar;
ylabel(hb,'Mahalanobis Distance')
legend('Sample Data from P.D.F. surface','Testing points','Location','NW');
end
```

7.4 Script for assignment 5, exercise 2, solve a quadratic equations system analytically and plotting the decision boundary between two classes given a certain prior probability, the mean and the covariance

```
function assignment5ex2()
    syms x1 x2
    g1=(-1*((x1-3)^2)/2-(((x2-5)^2)/8)+log(0.15));
    g2=(-1*((x1-2)^2)/4)-(((x2-1)^2)/2)+log(0.7)-log(sqrt(2));

    %Solve the quadratic equations system
    [x1, x2] = solve([g1-g2==0], [x1, x2]);
    x1=real(x1)
    disp(x2)

    %Plot the decision boundary
    ezplot((g1-g2))
    title('Decision Boundary')
end
```

7.5 Program in C++ and script in Matlab for assignment 4

7.5.1 Program in C++ for the simulation

```
#include <iostream>
```

```cpp
#include <cstdio>
#include <cstring>
#include <stdlib.h>     /* srand, rand */
#include <time.h>       /* time */
using namespace std;

typedef unsigned int uint;

const uint N_PEOPLE = 1000000;
const uint N_TURNS = 100;

uint endpoints[N_TURNS];

int main() {
    memset(endpoints, 0, N_TURNS); // Set all initial positions to 0

    srand(time(NULL));

    // Every player plays 100 turns
    for (uint player = 0; player < N_PEOPLE; ++player) {
        uint current_endpoint = 0;

        for (uint turn = 0; turn < N_TURNS; ++turn)
            current_endpoint += rand() & 1; // Add random bit

        ++endpoints[current_endpoint]; // Increment the # of ppl ending here
    }

    for (uint turn = 0; turn < N_TURNS; ++turn)
        cout << endpoints[turn] << endl;
}
```

7.5.2 Matlab script

```
% randomwalk.dat is a file with the output of the C++ program
fileID = fopen('randomwalk.dat', 'r');
data = fscanf(fileID, '%d');
plot(data);
```