Lab 1

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

Pattern Recognition

FMNS • RUG

November 23, 2017

1 Assignment 1

1.1 Exercise 1

We compute the matrix of correlation coeficients between the data variable. Note that cell i-j indicates the correlation coefficient between variable i and j, where the indices are the columns of the variables in the data matrix.

Command: `corrcoef(lab1_1)`

Result:

1.0000	-0.0615	0.7156
-0.0615	1.0000	0.5142
0.7156	0.5142	1.0000

As we can see, the matrix is symettric and has one's in the diagonal, which makes sense because the correlation is symettric and the values of the diagonal indicate correlation between same variables, which are always 1. Negative correlations indicate a inverse proportionality between the variables.

1.2 Exercise 2

- **Plot A**:
  Command: `A = scatter(lab1_1(:,1),lab1_1(:,3))`

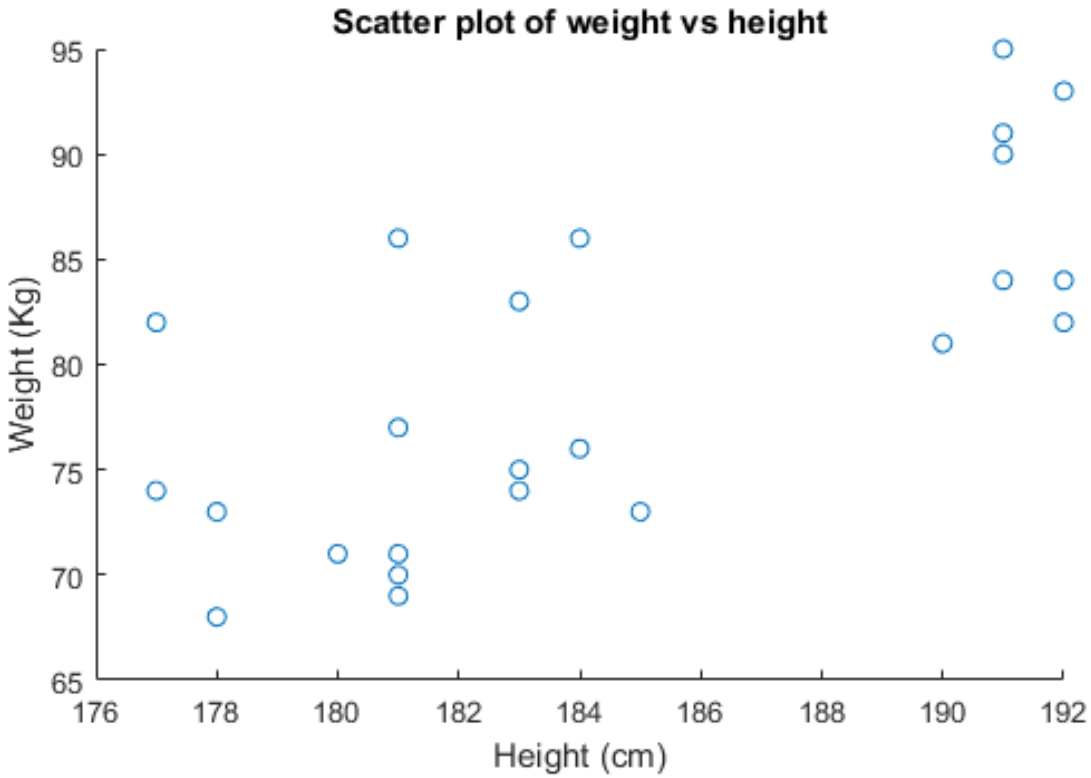

Figure 1: Relation between weight and height

- **Plot B**:
  Command: `B = scatter(lab1_1(:,2),lab1_1(:,3))`

Figure 2: Relation between weight and age

According to the correlation coefficients and the plots (1, 2) we can conclude that with both pairs of variables there is a direct relationship between the two variables with a positive correlation tendency, although the correlation is stronger in PlotA (1) (0.7156) as compared to PlotB (2) (0.5142). So we could say, according to the available data, that in plot A the weight variable is directly proportional to te height variable. On the other hand, in plot B, the correlation seems to be negative in people aged from 20 to 37 years, and positive from 38 to 55 years, but we cannot come to a definite conclusion because we don't have enough sample data.

2 Assignment 2

2.1 Exercise 1

We compute *Set S* and *Set D* with two scripts (Figures 3.1 and 3.2) which follow the algorithm described in the statement. We use `Matlab`'s `pdist` method with 'hamming' metric, in order to get the normalized hamming distance between the pair of iris codes in each iteration for both sets.

2.2 Exercise 2

We then plot the histograms of Hamming distances for both sets on the same figure (3) using 3.3, and we can observe that there is no overlap between the 2 histograms, although this depends on the sampling (we ran the script many times and got small overlaps sometimes). The low overlap is what allows us to clearly distinguish between same and different person iris codes, with a very low error rate.

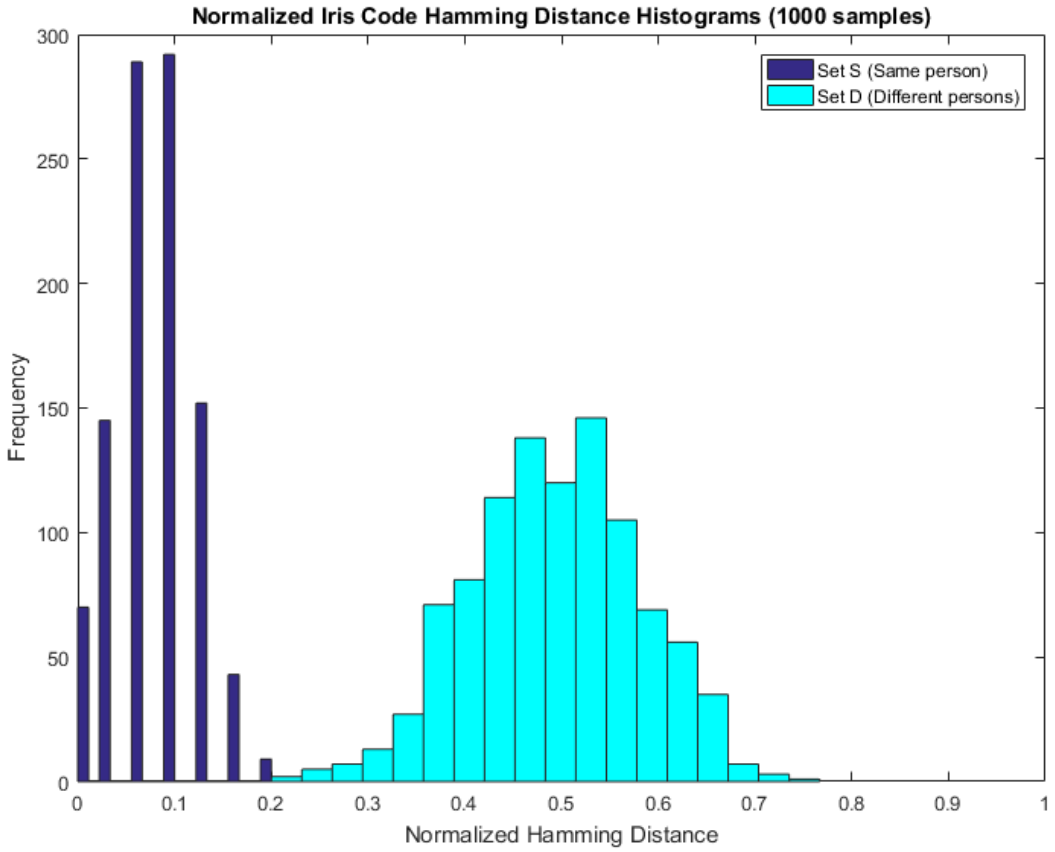

Figure 3: Normalized hamming distance frequency histogram for Set S and Set D

2.3 Exercise 3

We compute the mean and variances using Matlab functions `mean` and `var` for each set (See 3.4, lines 19-23)).

(a)

$$\overline{X}_S = 0.0825 \tag{1}$$

$$\sigma_S^2 = 0.0018 \tag{2}$$

$$\overline{X}_D = 0.4940 \tag{3}$$

$$\sigma_D^2 = 0.0087 \tag{4}$$

As we can see, the mean is much higher for different person's iris codes, which tells us that its HD is tipically much higher, and thus their iris codes differ in much more bits, allowing us to discern. Also the variance is higher for different person's iris codes, because their iris codes bits are statistically independent and thus can have more variability in the hamming distance, if they happen to have more or less similar bits by chance.

(b) We can estimate the probability of two same-position iris code bits from different persons being different by looking at the normalized HD mean for *Set D*. The HD for this set tells us, based on the measured samples, what is the proportion of same-position bits of iris codes from different persons that differ. If we take the mean of all the samples of normalized HD values, we can get a good approximation of this probability as the mentioned proportion, so as indicated in `(a)` the probability is $\overline{X}_D = 0.4979$.

(c) Again, we can estimate the number of statistically independent bits using the Hamming Distance test. $\overline{X}_S$ and $\overline{X}_D$ tell the proportion of statistically independent bits for same person iris codes and different person iris codes respectively, so we only need to multiply it by the number of bits in an iris code, in this case 30. So, based on our samples, for *Set S* we expect $\overline{X}_S * 30 = 2.475$ bits to be different and for *Set D* we expect $\overline{X}_D * 30 = 14.82$ bits to be different, which makes sense because different person's iris codes are expected to have much more differing bits than same person's iris codes.

2.4 Exercise 4

We generate the plot (Figure 4) with the normal distributions using Matlab's `normpdf` and `plot` functions (See 3.4). Because the normpdf function generates a gaussian function whose area below the curve is equal to 1, if we want it to fit our bar histogram we need to scale it appropriately. A possible way is to multiply the normal distribution by the area of the whole histogram, so that now the area below the curve happens to be exactly the area of the histogram. In order to compute the area of the histogram, we compute the width of each bar and multiply by the number of samples (1000 in this case). In the case of the *Set S* histogram, we also multiply the computed area by 3 because it has several blank bars where no value falls into, but which still occupy space.

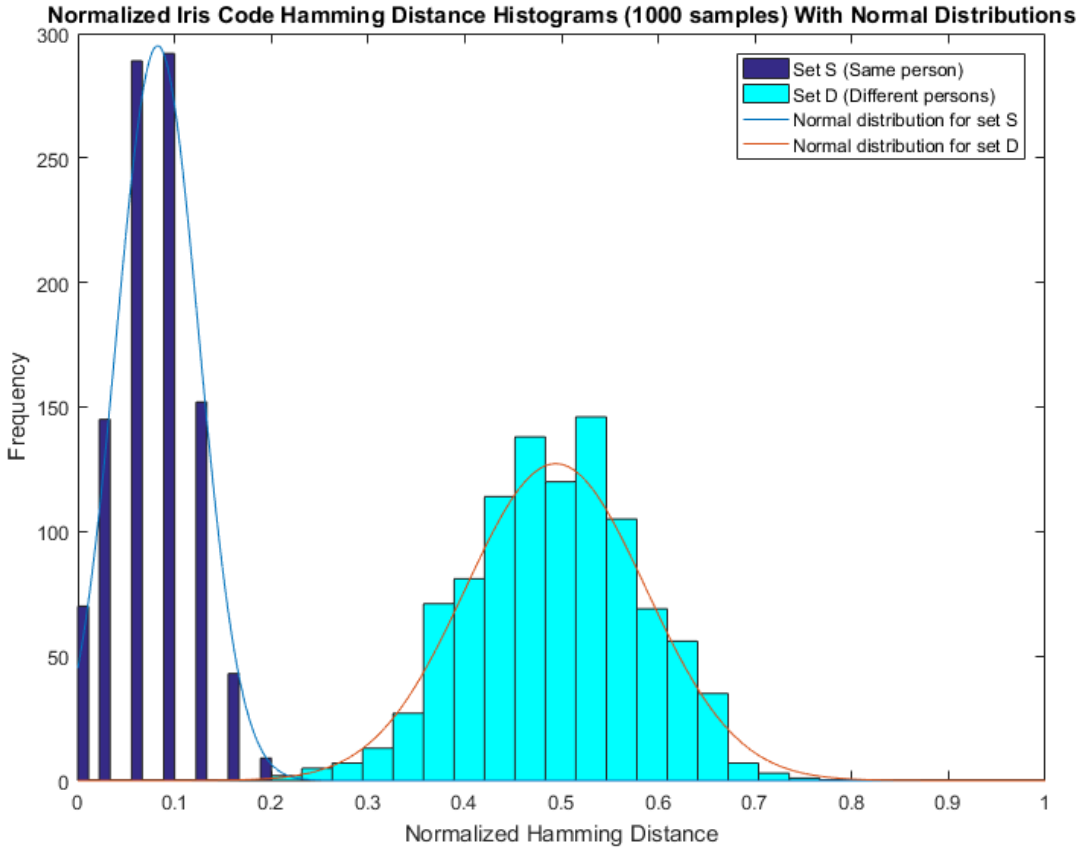

Figure 4: Normalized hamming distance frequency histogram for Set S and Set D with normal distribution plots

2.5 Exercise 5

(a) For a false acceptance error of 0.0005, we can estimate the decision criterion value by using the `erfinv` Matlab's method with a confidence interval of $1 - 0.0005 * 2$, in order to compute the critical Z value that we will then adjust to our distribution with our mean and stdev. We need to double the acceptance error because the confidence interval leaves two tails, one on each side on the distribution, and we want each of them to be 0.00005 wide (although we are only interested in the left one).

Commands:

(i) `confidence = 1 - 0.0005*2`

(ii) `normz = -sqrt(2)*erfinv(confidence)`

(iii) `ourdistrz = normz*std(hdd) + mean`

Also, we give a negative sign to `normz`, because we are interested in the left side's `z` of the distribution.

Result: HD = 0.1872 is the value of the decision criterion

This result makes sense if we look at the normal distribution plot for *Set D* in Figure **??**, as we can see that the area below the curve for HD ¡ 0.1872 is very low, just as 0.0005 is.

(b) Now, in order to determine the false rejection rate, we only need to compute the integral of the normal distribution for *Set S* with $HD > 0.1872$. We can do that with Matlab's `normcdf` function, using the mean and stdev of the distribution of *Set S*. This gives us the value of the integral for $HD <= 0.1872$, so in order to get the opposite area of the distribution, we will have to substract it from 1.

Command: `1 - normcdf(0.1872, mean(hds), std(hds))`

Result: False rejection rate = **0.0070**

This rate is obviously higher than the false acceptance rate because again in in Figure **??** we see that the area below the curve of the gaussian function for *Set S* and HD ¿ 0.1872 is larger than the area of the previous question. Also makes sense because we always prefer to be more accurate when affirming an acceptance rather than affirming a rejection.

2.6 Exercise 6

We compute the most likely person with the script provided in the appendix (3.5), where the algorithm is explained with comments.

The basic idea is to compute the average normalized HD from testperson to every `person01...person20` and take the one with the lowest average HD. Of course we need to compute the HD without considering unknown bits. We can do that easily by computing the HD normally and then substracting the contributed amount of the unknown bits, which will always be the same because they will always be different. In this case with 10 unknown bits we need to substract 10/30, (notice that we divide by 30 because the HD is normalized). We also need to scale the HD to normalize only over the known bits, multiplying by 30/20.

Results:

- **Most likely person:** 5
- **Significance level:** 1.1401e−4

As we can see from the results, the significance level convinces us that testperson's iris code belongs to person05 with a very high probability ($1 - significance = 0,9998599$).

3 Appendix

3.1 Script to compute *Set S*

```
function hds = hd_set_s()
    % Generate array with all person files
    personFiles = [];
    for i = 1:20
      personFiles(i, :) = sprintf('person%02d.mat', i);
    end

    for time = 1:1000
      randomPerson = char(personFiles(randi(20), :)); % Select random person
      load(randomPerson);

      % Select two random iris codes from the same person
      rowA = iriscode(randi(20), :);
      rowB = iriscode(randi(20), :);

      % Store normalized hamming distance between both rows in the array hds
      hds(time) = pdist([rowA; rowB], 'hamming');
    end
```

3.2 Script to compute *Set D*

```
function hdd = hd_set_d()
    % Generate array of all person files
    personFiles = [];
    for i = 1:20
      personFiles(i, :) = sprintf('person%02d.mat', i);
    end

    for time = 1:1000
      % Select two different random persons
      randomindexes = randperm(20, 2);

      randomPersonA = char(personFiles(randomindexes(1), :));
      randomPersonB = char(personFiles(randomindexes(2), :));

      % Select a random iris code from the first person
      load(randomPersonA);
      rowA = iriscode(randi(20), :);

      % Select a random iris code from the second person
      load(randomPersonB);
      rowB = iriscode(randi(20), :);

      % Store normalized hamming distance between both iris codes in the array hds
      hdd(time) = pdist([rowA; rowB], 'hamming');
    end
```

3.3 Script to generate the histogram plot

```
function plotHistograms(hds, hdd, nbins)
    % Generate histograms from each set with the specified number of bins
    [n, x] = hist(hds, nbins);
    [n2, x2] = hist(hdd, nbins);

    % Plot the first histogram
    bar(x, n, 'hist');
    hold on;

    % Plot the second histogram
    h = bar(x2, n2, 'hist');

    % Define x-axis limits
    xlim([0 1]);

    title('Normalized Iris Code Hamming Distance Histograms (1000 samples)');
    xlabel('Normalized Hamming Distance');
    ylabel('Frequency');
    hold off;

    set(h, 'facecolor', 'c'); % Change color for Set D plot

    legend('Set S (Same person)', 'Set D (Different persons)');
```

3.4 Script to generate the histogram plot with fitted normal distribution and compute statistics

```
function [ms,vs,md,vd] = plotHistogramsDistribution(hds, hdd, nbins)
    % Generate histograms from each set with the specified number of bins
    [n, x] = hist(hds, nbins);
    [n2, x2] = hist(hdd, nbins);

    % Plot the first histogram
    bar(x, n, 'hist');
    hold on;

    % Plot the second histogram
    h = bar(x2, n2, 'hist');

    title('Normalized Iris Code Hamming Distance Histograms (1000 samples) With Normal
        Distributions');
    xlabel('Normalized Hamming Distance');
    ylabel('Frequency');

    set(h, 'facecolor', 'c'); % Change color for Set D plot

    % Compute mean and variance (statistics)
    ms = mean(hds);
    vs = var(hds);
    md = mean(hdd);
    vd = var(hdd);

    % Generate normal distributions with the respective means and stdevs
    norms = normpdf([0:0.00001:1], ms, std(hds));
    normd = normpdf([0:0.00001:1], md, std(hdd));

    % Plot the normal distributions, scaling them by the area of the histogram bins
```

```
    % x(nbins) - x(1) gives us the width of all the bins, thus dividing by nbins we get
    % the width of each bin. We then multiply by the number of samples (1000) to get
    % the area, and in the case of Set S we also add a factor of 3 to account for
    % the blank bins.
    plot([0:0.00001:1],norms*1000*(x(nbins) - x(1))*3/nbins);
    plot([0:0.00001:1],normd*1000*(x2(nbins) - x2(1))/nbins);

    legend('Set S (Same person)', 'Set D (Different persons)', 'Normal distribution for set
        S', 'Normal distribution for set D');

    hold off;
```

3.5 Script to compute testperson most likely match and its significance

```
function [minimumperson, significance] = whoistest()
    load testperson;
    testiriscode = iriscode(1,:); % Save test person iris code

    % Initialize minimum hd value and minimum persons
    % 1 is okay  because it's the maximum possible hd
    % Initially there is no person, we indicate that with 0
    minimumperson = 0;
    HDmin = 1;

    % Initialize matrix that will contain all HD's for significance analysis
    hdseveryone = zeros(20, 20);

    % For every person we compute the average normalized Hamming Distance to
    % testperson, without considering bits with value 2
    for person = 1:20
        hdtestperson = zeros(20, 1); % Contains all the HDs of person with test
        personfile = sprintf('person%02d.mat', person);
        load(personfile);

        % Compute HD for every iris code of the person
        for code = 1:20
            % Because there are 10 unknown values (2's) in testperson, we need to substract
            % 10/30 from the normalized HD, and scale the result multiplying by 30/20
            % so that the HD is normalized only over the known bits (20)
            hdtestperson(code) = (pdist([iriscode(code, :); testiriscode], 'hamming') -
                10/30)*(30/20);
        end

        % Add the HDs to the matrix of everyone's HD that we will use for
        % significance analysis
        hdseveryone(person, :) = hdtestperson;
        avghd = mean(hdtestperson);

        % Update minimum variables accordingly if necessary
        if avghd < HDmin
            HDmin = avghd;
            minimumperson = person;
        end
    end

    % Eliminate the minimum person HDs from the matrix of every HD, because we only
    % want to compute the distribution function for HDs of persons different
    % than testperson
    hdseveryone(minimumperson, :) = [];
    hdseveryone = hdseveryone(:); % See matrix as array of values
```

```
46      % Compute significance as integral of the distribution where HD <= HDmin
47      significance = normcdf(HDmin, mean(hdseveryone), std(hdseveryone));
```