Assignment 2

Maks Epsteins

October 19, 2021

Exercise 1

$f_1 = C$: The convolution of the image is conducted with respect to the y-coordinate.

$f_2 = A$: The convolution of the image is conducted with respect to the x-coordinate.

$f_3 = E$: This is a so called box blur kernel, since it averages each value in a box, in this case a 3x3 box.

$f_4 = D$: This kernel corresponds to an edge detection filter which emphasises pixels where there are big differences in contrast.

$f_5 = B$: This convolution kernel subtracts all pixels values on the upper right diagonal and above and adds the pixel values below said diagonal. This results in high contrasts regions in areas with diagonal edges that are angled in the same way as the convolution kernel.

Exercise 2

a)

Linear interpolation is the process of curve fitting data points with linear polynomials. $F_{lin}(x)$ is continuous within the specified range but not differentiable due to the fact that the curve does not have a defined derivative at the data points.

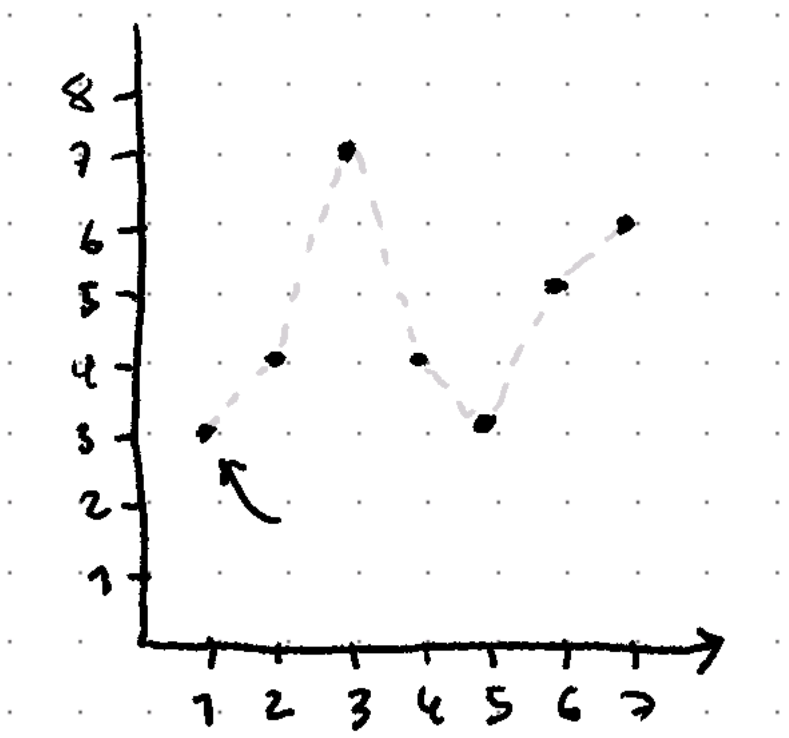


Figure 1: Plot of f

b)

The function g(x) which corresponds to linear interpolation is the triangular function:

$g(x) = max(1 - |x|, 0)$

c)

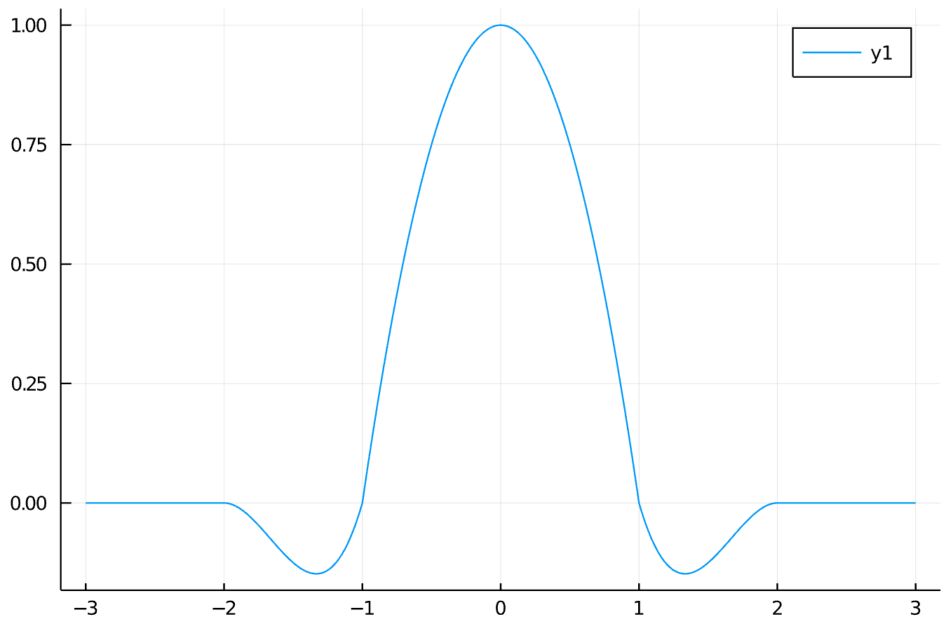


Figure 2: Plot of g(x)

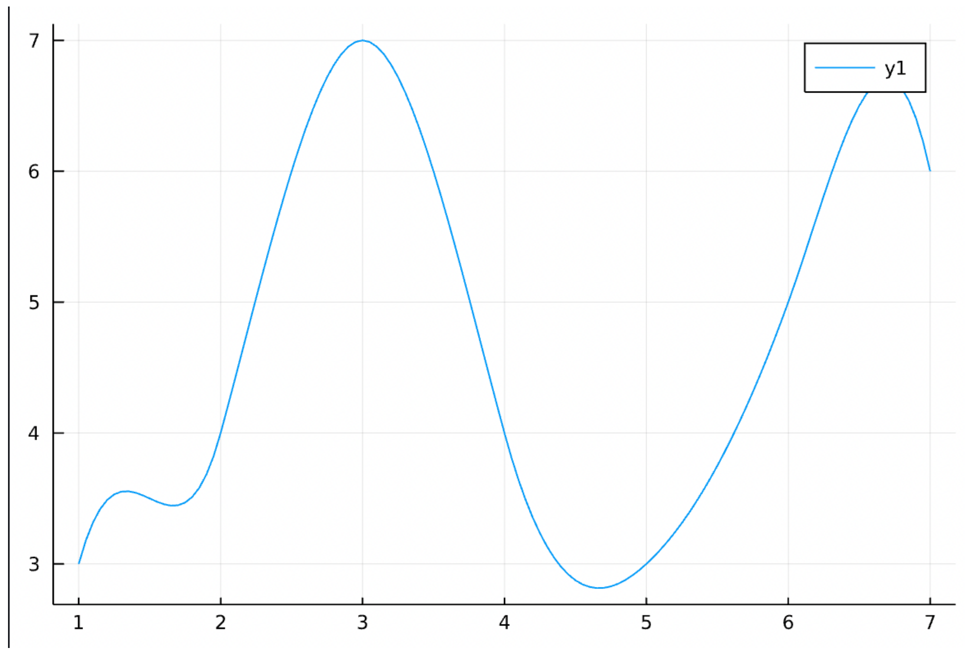


Figure 3: Plot of $F_g(x)$

$F_g(x)$ appears to be both continuous and differentiable. $F_g(x)$ is differentiable if the function g(x) is since $F_g(x)$ is a linear sum of g(x-i) functions. Therefore we need to first prove that g(x) is differentiable in order to then prove that $F_g(x)$ is. In order to prove this we need to prove that the derivative of g(x) is defined in the points x = 0,1,2 since those are the points where g(x) changes functions. To prove this we will look at the limits when x approaches 0,1,2 from both the negative and positive side of the function $g'(x)$.

x -> 0:

$lim_{x \to 0+} g'(x) = lim_{x \to 0-} g'(x)$ since g'(x) is symmetrical around the point x = 0. => g(x) is differentiable in the point x = 0.

x -> 1:
x -> 1-:
$lim_{x \to 1-} g'(x) = lim_{x \to 1-} \frac{d}{dx}(|x|^3 - 2|x|^2 + 1)$
=> since x > 0:
$lim_{x \to 1-} \frac{d}{dx}(x^3 - 2x^2 + 1) = lim_{x \to 1-} 3x^2 - 4x = 3 \cdot (1)^2 - 4 \cdot (1) = -1$

x -> 1+:
$lim_{x \to 1+} g'(x) = lim_{x \to 1+} \frac{d}{dx}(-|x|^3 + 5|x|^2 - 8|x| + 4)$
=> since x > 0: $lim_{x \to 1+} \frac{d}{dx}(-|x|^3 + 5|x|^2 - 8|x| + 4) = lim_{x \to 1+} \frac{d}{dx}(-x^3 + 5x^2 - 8x + 4) = lim_{x \to 1+}(-3x^2 + 10x - 8) = -1$

Since $lim_{x \to 1+} g'(x) = lim_{x \to 1-} g'(x)$ then g(x) is differentiable in said point.

x -> 2:

x -> 2-:

$lim_{x \to 2-} \frac{d}{dx}(-|x|^3 + 5|x|^2 - 8|x| + 4)$

=> since x > 0:

$lim_{x \to 2-} \frac{d}{dx}(-x^3 + 5x^2 - 8x + 4) = lim_{x \to 2-}(-3x^2 + 10x - 8) = 0$

x -> 2+:

g(x) = 0 if x > 2 => g'(x) = 0 => therefore $lim_{x \to 2-} g'(x) = lim_{x \to 2+} g'(x)$ => g(x) is differentiable in x = 2.

From these calculations we have proved that g(x) is differentiable for all points within the range of definition and therefore $F_g(x)$ is also differentiable.

Exercise 3

3.1 Nearest neighbors

3.1.1 Example of Calculations

For example if we start the data point 0.4010. We look through all the training data points and calculate the absolute value of each training data point minus the data point we are testing. Then we look at which point gives us the smallest result and that is our nearest neighbour which for the point 0.4010 corresponds to 0.4003. We then check which class 0.4003 corresponds to, which is class 1 and then we proceed with classifying our point 0.4010 in to the class of its nearest neighbour => 0.4010 is classified as class 1.

3.1.2 Table with Classifications

Data Point	Nearest Neighbour	Classified as	Actual Class
0.4010	0.4003	1	1
0.3995	0.3998	1	1
0.3991	0.3997	1	1
0.3287	0.3139	2	2
0.3160	0.3139	2	2
0.2924	0.3139	2	2
0.4243	0.4003	3	1
0.5005	0.5632	3	3
0.6769	0.7586	3	3

3.2 Gaussian Classification

3.2.1 Example of calculations

For example if we start with the point 0.4003. We would calculate the likelihood of our point appearing in the distributions for each class by for example using the normpdf function in matlab. Then we take the likelihood which is the highest and classify our point to the class which said likelihood corresponds to. So the highest likelihood for the point 0.4003 is for class 1 and therefore the point is classified into the said point.

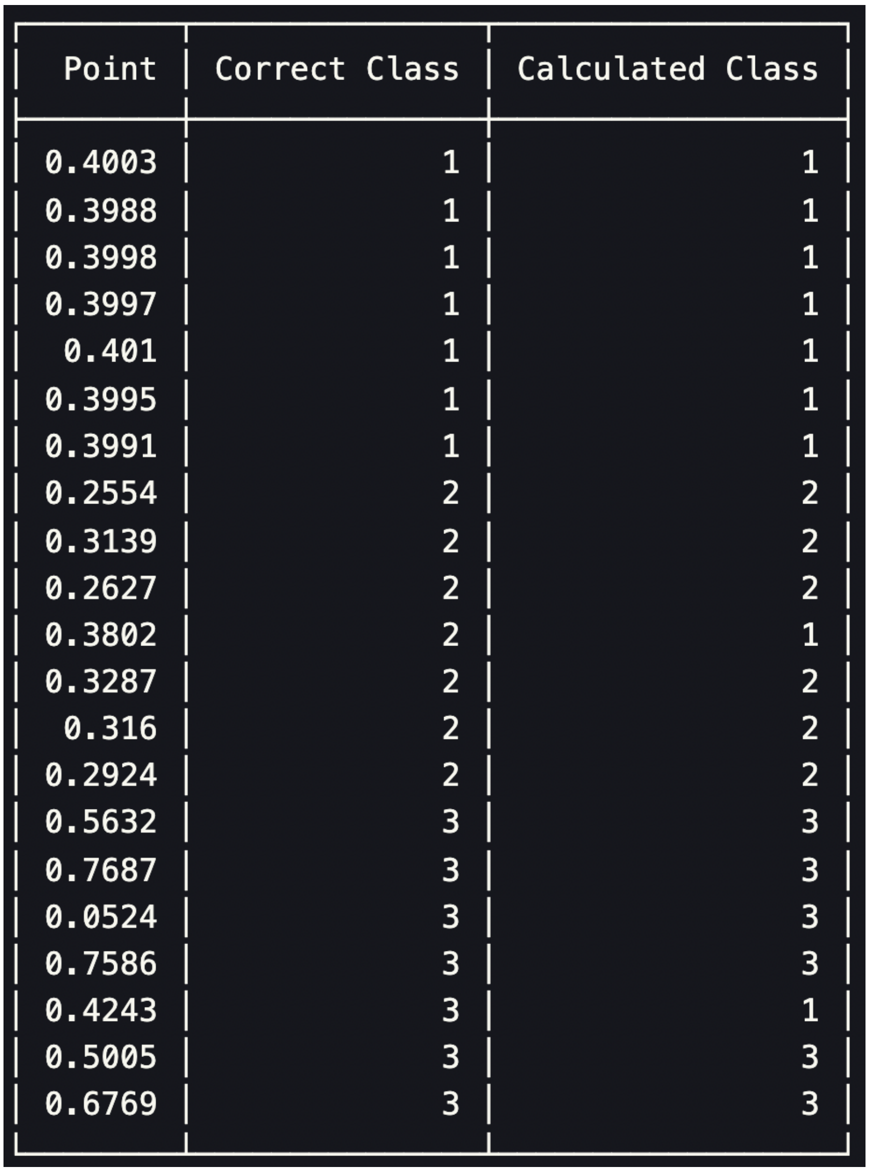
3.2.2 Code

```
%Dictionary where each data point is saved together with
%its class
classes = Dict(
    [1 => [0.4003,0.3988,0.3998,0.3997,0.4010,0.3995,0.3991],
     2 => [0.2554,0.3139,0.2627,0.3802,0.3287,0.3160,0.2924],
     3 => [0.5632,0.7687,0.0524,0.7586,0.4243,0.5005,0.6769]]
)

function gaussian_classification()

    %The provided distributions for each class
```

```
d1 = Normal(0.4,0.01)
d2 = Normal(0.3,0.05)
d3 = Normal(0.5,0.2)

pcopy = []
correctclass = []
actualclass = []
%Loop through the points for each class
for n in 1:3
    points = classes[n]
    for p in points
        %Adds points to "Point" column in table

        push!(pcopy,p)

        %Adds the class in which the point is supposed to
        %classified

        push!(correctclass,n)

        %Calculates the likelihood of the point appearing
        %in each normal distribution.
        %pdf in julia is equivalent to normpdf in matlab.

        l1,l2,l3 = pdf(d1, p), pdf(d2, p), pdf(d3, p)

        %Finds the index of the maximum value in the
        %provided array. Julia is 1 indexed so the index
        %will be the same as the correct class.

        max = findmax([l1,l2,l3])[2]

        %We push the calculated class to its array

        push!(actualclass,max)
    end
end

%We concatenate our data along dimension 2 to be able to
%construct a table

data = hcat(pcopy,correctclass,actualclass)

%Initialize header variable for the table
header = (["Point", "Correct Class", "Calculated Class"])

%Returns the table that can be seen below at 3.2.2
```

```
    return pretty_table(data, header)
end

gaussian_classification()
```

3.2.3 Result



```
┌────────┬───────────────┬──────────────────┐
│  Point │ Correct Class │ Calculated Class │
├────────┼───────────────┼──────────────────┤
│ 0.4003 │             1 │                1 │
│ 0.3988 │             1 │                1 │
│ 0.3998 │             1 │                1 │
│ 0.3997 │             1 │                1 │
│  0.401 │             1 │                1 │
│ 0.3995 │             1 │                1 │
│ 0.3991 │             1 │                1 │
│ 0.2554 │             2 │                2 │
│ 0.3139 │             2 │                2 │
│ 0.2627 │             2 │                2 │
│ 0.3802 │             2 │                1 │
│ 0.3287 │             2 │                2 │
│  0.316 │             2 │                2 │
│ 0.2924 │             2 │                2 │
│ 0.5632 │             3 │                3 │
│ 0.7687 │             3 │                3 │
│ 0.0524 │             3 │                3 │
│ 0.7586 │             3 │                3 │
│ 0.4243 │             3 │                1 │
│ 0.5005 │             3 │                3 │
│ 0.6769 │             3 │                3 │
└────────┴───────────────┴──────────────────┘
```

Figure 4: Caption

We are able to use the likelihoods of each data point appearing in each class to classify our points instead of MAP since the a priori probabilities of a point appearing in each class are assumed to be equivalent. As can be seen in the table all of the data points are correctly classified except for two, the points p = 0.4243 and p = 0.3802.

Exercise 4

The matrices A_k denote the three possible matrices, I denotes the distorted matrix.

$$A_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, A_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, A_3 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, I = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

The probabilities for each matrix A_k:

$$P(A_1) = 1/4, P(A_2) = 1/2, P(A_3) = 1/4$$

Expressions for each matrix A_k given image I:

$$P(A_1|I) = \frac{P(I|A_1) \cdot P(A_1)}{P(I)} = \frac{\frac{\varepsilon}{16}}{P(I)}, P(A_2|I) = \frac{P(I|A_2) \cdot P(A_2)}{P(I)} = \frac{\frac{\varepsilon^3}{4}}{P(I)}, P(A_3|I) = \frac{P(I|A_3) \cdot P(A_3)}{P(I)} = \frac{\frac{\varepsilon^2}{16}}{P(I)}$$

$$P(A_1|I) + P(A_2|I) + P(A_3|I) = 1$$

a) $\varepsilon = 0.05$:

$$P(A_1|I) = \frac{0.95^3 \cdot 0.05 \cdot 0.25}{P(I)} = \frac{0.0107171875}{P(I)}, P(A_2|I) = \frac{0.05^3 \cdot 0.95 \cdot 0.5}{P(I)} = \frac{0.0011875}{P(I)}, P(A_3|I) = \frac{0.05^2 \cdot 0.95^2 \cdot 0.25}{P(I)} = \frac{0.0005640625}{P(I)}$$

$$=>$$

From adding all the probabilities we get P(I) = 0.011340625.

The maximum a posteriori is for matrix A_1:

$$\frac{0.0107171875}{0.011340625} \approx 94.5\%$$

b) $\varepsilon = 0.5$:

$$P(A_1|I) = \frac{0.5^4 \cdot 0.25}{P(I)} = \frac{0.015625}{P(I)}, P(A_2|I) = \frac{0.5^4 \cdot 0.5}{P(I)} = \frac{0.03125}{P(I)}, P(A_3|I) = \frac{0.5^4 \cdot 0.25}{P(I)} == \frac{0.015625}{P(I)}$$

$$=>$$

From adding all the probabilities up to 1 we get that P(I) = 0.0625 and therefore =>

The maximum a posteriori is for matrix A_2:

$$\frac{0.03125}{0.0625} = 0.5 = 50\%$$

Exercise 5

Correct image matrices denoted A_k where k is the column consisting of black pixels:

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, A_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, A_3 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, A_4 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The probability of occurrence for each matrix A_k:

$$P(A_1) = P(A_4) = 0.3, P(A_2) = P(A_3) = 0.2$$

Image matrix with errors denoted I:

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Probability of getting A_k given image I:

$$P(A_1|I) = \frac{P(I_1) \cdot P(A_1)}{P(I)} = \frac{(1-\varepsilon)^{10} \cdot \varepsilon^6 \cdot 0.3}{P(I)} \approx 2.06158 \cdot 10^{-}6,$$

$$\mathrm{P}(\mathrm{A}_2|I) = \frac{P(I_2) \cdot P(A_2)}{P(I)} = \frac{(1-\varepsilon)^{12} \cdot \varepsilon^4 \cdot 0.2}{P(I)} \approx 2.19902 \cdot 10^{-}5,$$

$$\mathrm{P}(\mathrm{A}_3|I) = \frac{P(I|A_3) \cdot P(A_3)}{P(I)} = \frac{(1-\varepsilon)^{10} \cdot \varepsilon^6 \cdot 0.2}{P(I)} \approx 1.37439 \cdot 10^{-}6,$$

$$\mathrm{P}(\mathrm{A}_4|I) = \frac{P(I_4) \cdot P(A_4)}{P(I)} = \frac{(1-\varepsilon)^8 \cdot \varepsilon^8 \cdot 0.3}{P(I)} \approx 1.28849 \cdot 10^{-}7$$

$$=>$$

The maximum a posteriori is for matrix A_2:

$$\frac{P(A_2|I)}{\sum_{k=1}^{4} P(A_k|I)} \approx 0.8605 \approx 86\%$$

Exercise 6

$$\omega_1 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \omega_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \omega_3 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$x = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

$\mathrm{P}(\omega_1) = 0.25, P(\omega_2) = 0.4, P(\omega_3) = 0.35$

P(measured black | white) = 0.35, P(measured white | black) = 0.25, P(measured black | black) = 0.75, P(measured white | white) = 0.65

Calculation of a posteriori probabilities for each matrix ω_k

ω_1:

$\mathrm{P}(\omega_1|x) = \frac{P(x|\omega_1)\cdot P(\omega_1)}{P(x)} = 0.25^5 \cdot 0.65^5 \cdot 0.75^5 \cdot 0.25/P(x) \approx 6.7222 \cdot 10^-6/P(x)$

ω_2:

$\mathrm{P}(\omega_2|x) = \frac{P(x|\omega_2)\cdot P(\omega_2)}{P(x)} = 0.25^5 \cdot 0.65^5 \cdot 0.75^3 \cdot 0.25^2 \cdot 0.4/P(x) \approx 2.3423 \cdot 10^-6/P(x)$

ω_3:

$\mathrm{P}(\omega_3|x) = \frac{P(x|\omega_3)\cdot P(\omega_3)}{P(x)} = 0.65^7 \cdot 0.25^3 \cdot 0.75^4 \cdot 0.35/P(x) \approx 2.96889 \cdot 10^-5/P(x)$

The maximum a posteriori is for matrix A_2:

$$\frac{P(\omega_3|x)}{\sum_{k=1}^{3} P(\omega_k|I)} \approx 0.7661 \approx 76\%$$

Exercise 7

7.1 Matlab code

```
function features = segment2features(I)
% features = segment2features(I)

%imshow(I);

[rows,cols] = size(I);

top = [0,0];
right = [0,0];
bot = [0,0];
left = [0,0];

for r = 1:rows
    for c = 1:cols
        if I(r,c) > 0
            bot = [r,c];
            if top == [0,0]
                top = [r,c];
            end
            if left == [0,0]
                left = [r,c];
            end
            if c < left(2)
                left = [r,c];
            end
            if right == [0,0]
                right = [r,c];
            end
            if right(2) < c
                right = [r,c];
            end
        end
    end
end
top = top(1) - 1;
bot = bot(1) + 1;
right = right(2) + 1;
left = left(2) - 1;
number = zeros(abs(top-bot), abs(right-left));

for r = top:bot
    for c = left:right
        if I(r,c) ~= 0
            number(r-top+1,c-left+1) = 1;
        else
            number(r-top+1,c-left+1) = 0;
        end

    end
end

[height, width] = size(number);
x_mid = round(width/2);
y_mid = round(height/2);

%Nbr of holes in the image, 1 is subtracted to not count the background
nbrholes = max(max(bwlabel(not(number)))) - 1;

%Density of white pixels in the image
pixeldensity = sum(sum(number)) / numel(number);

%The amount of 1s in the upper and lower half of the cropped image.
upperhalf = number(1:y_mid,:);
lowerhalf = number(y_mid:end,:);

upperhalfdensity = sum(sum(upperhalf)) / numel(upperhalf);
lowerhalfdensity = sum(sum(lowerhalf)) / numel(lowerhalf);

proportionaldensity = 5 * (upperhalfdensity - lowerhalfdensity);

%Calculating the longest column consisting of 1s in the cropped image.

longestCol = 1/max(sum(number));


%The center of mass of the image both in the x and y coordinates and their distance from the
center_of_mass = regionprops(number, 'Centroid');
center_of_mass = center_of_mass.Centroid - [x_mid,y_mid];

features = [nbrholes,longestCol,center_of_mass(1),
center_of_mass(2),proportionaldensity,pixeldensity];
```

7.2 Plot of the results

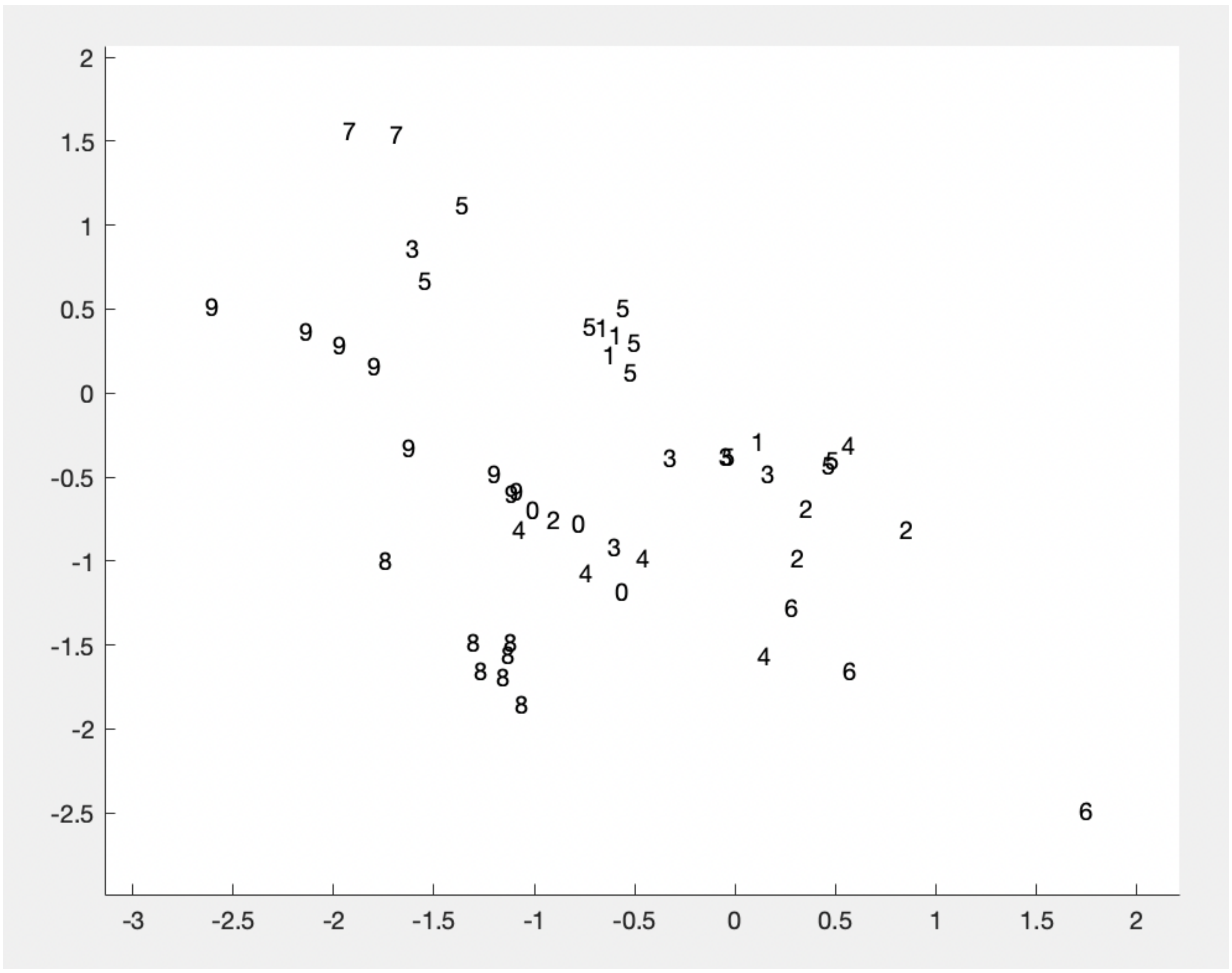


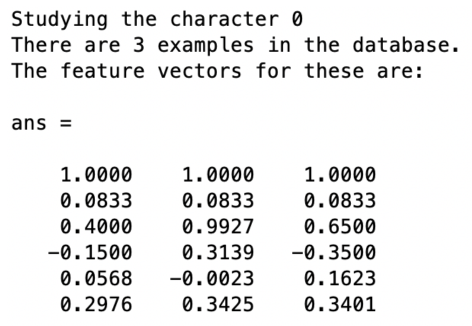
Figure 5: Plot of all the numbers

It is starting to resemble some sort of feature extraction however it is not quite up to standard. Some of them are sorted pretty well others numbers are quite chaotic. Have tried many different things but haven't gotten it to work well so am going to ask about it during the next exercise session.

Update:

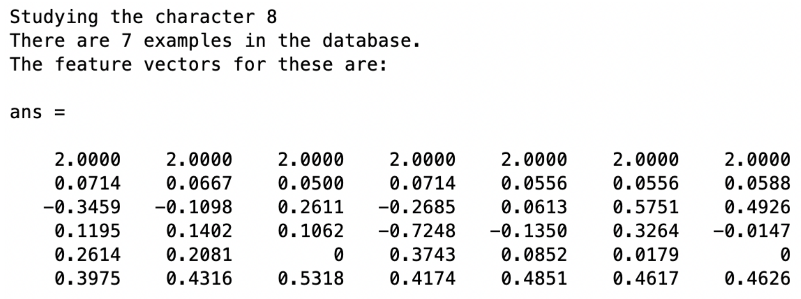
So after doing assignment 3 I made a lot of improvements to the feature extraction and therefore I did not update these results since I did not know that that would be part of the next exercise.

7.3 Results for 0 and 8 as Examples



```
Studying the character 0
There are 3 examples in the database.
The feature vectors for these are:

ans =

    1.0000    1.0000    1.0000
    0.0833    0.0833    0.0833
    0.4000    0.9927    0.6500
   -0.1500    0.3139   -0.3500
    0.0568   -0.0023    0.1623
    0.2976    0.3425    0.3401
```

Figure 6: Results for 0



```
Studying the character 8
There are 7 examples in the database.
The feature vectors for these are:

ans =

    2.0000    2.0000    2.0000    2.0000    2.0000    2.0000    2.0000
    0.0714    0.0667    0.0500    0.0714    0.0556    0.0556    0.0588
   -0.3459   -0.1098    0.2611   -0.2685    0.0613    0.5751    0.4926
    0.1195    0.1402    0.1062   -0.7248   -0.1350    0.3264   -0.0147
    0.2614    0.2081         0    0.3743    0.0852    0.0179         0
    0.3975    0.4316    0.5318    0.4174    0.4851    0.4617    0.4626
```

Figure 7: Results for 8