Assignment 2 - Image Analysis

Theodora Gaiceanu

1 Filtering

Filter f_1 corresponds to image C. Actually, the filter is the partial derivative of the image with respect to x. Consequently, the filter emphasis the vertical edges, as in figure C.
Filter f_2 corresponds to image A. In fact, the filter is the partial derivative of the image with respect to y. Therefore, the filter emphasis the horizontal edges, as in figure A.
Filter f_3 corresponds to image E. The filter used is an average filter, and convolving the image with it gives a blurred image.
Filter f_4 corresponds to image D. The filter used is a discrete Laplacian filter, having the purpose of finding all the edges by approximating the second derivatives with respect to x and y.
Filter f_5 corresponds to image B. The filter used emphasis the bigger parts of an image.

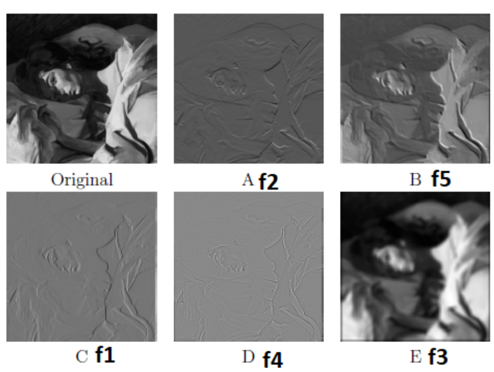

Figure 1: Corresponding filters for the images

2 Interpolation

1. In image analysis, interpolation is the process of transforming a discrete image f into a continuous image F. This can be illustrated in the following equation, given in [1]:

$$F_h(x, y) = I_h(f)(x, y) = \sum_{i=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} h(x - i, y - j) f(i, j) \tag{1}$$

From the mathematical point of view, linear interpolation makes use of linear polynomials in order to get new values within the range of a discrete set of points given as input. The linear interpolation sketch can be obtained by using the Matlab function *interp1* and plotting the result. The code is provided below.

```
f = [3 4 7 4 3 5 6];
x = (1:7)';
xi = (1:7)';
F = interp1(x,f,xi);
figure
plot(x,F,'o',xi,F)
title("Linear Interpolation")
```

Listing 1: Matlab code for 2.a

The linear interpolation function can be observed below.

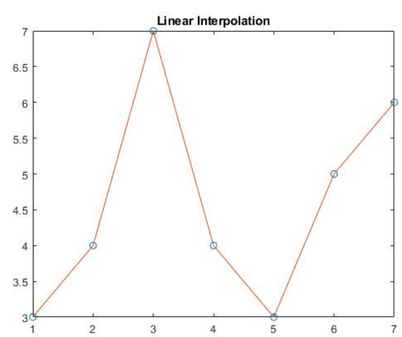

Figure 2: Linear interpolation

By analysing Figure 2, it can be seen that the function is defined in every point, therefore the function is continuous. But from its shape, one could say that the function is differentiable for certain intervals, but not for the whole interval [1, 7] (it has jumps on x = 2, 3, 4, 5, 6).

Also, from point (1,3) to point (2,4), the line has the equation $y = x+2$. The derivative of this is 1. From point (2,4) to point (3,7), the line has the equation $y = 3x - 2$. The derivative of this is 3. But $1 \neq 3$, so the left derivative in point (2,4) is not equal to the right derivative in the same point. Also, the derivatives in points $x = (2, 3, 4, 5, 6)$ can be computed with the Matlab function *diff*, which gives the following vector: $1, 3, -3, -1, 2, 1$. So, as $1 \neq 3$, $3 \neq -3$, $-3 \neq -1$, $-1 \neq 2$, $2 \neq 1$, one can say that the left derivatives in each points are not equal to the right derivatives in the same points, thus the function is not differentiable for the whole interval.

2. According to [1], the linear interpolation function has the form illustrated in Figure 3.

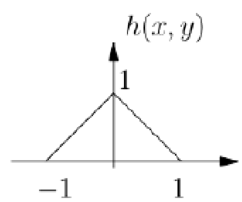

Figure 3: Linear interpolation function [1]

Therefore, the function *g(x)* has the following formulation:

$$g(x) = \begin{cases} 0, & \text{if } x < -1 \\ x+1, & \text{if -1 <= } x \text{ <= 0} \\ 1-x, & \text{if } 0 < x \text{ <= 1} \\ 0, & \text{if } 1 < x \end{cases} \tag{2}$$

This can also be written like:

$$g(x) = \begin{cases} 1-|x|, & \text{if } |x| \text{ <= 1} \\ 0, & \text{for the rest} \end{cases} \tag{3}$$

3. First, a function corresponding to *g(x)* has been implemented. This function takes x as input and gives as output the result. The code for *g(x)* is presented below.

```
function [func] = funct(x)
%g(x)
    if abs(x) <= 1
        func = abs(x)^3-2*abs(x)^2+1;
    else if abs(x) <= 2
        func = -abs(x)^3+5*abs(x)^2-8*abs(x)+4;
        else
            func = 0;
        end
    end
end
```

Listing 2: Matlab code for 2.c - g(x)

The function *g(x)* over the interval [-3, 3] is pictured in Figure 4.

Next, a function that defines the interpolation $F_g(x)$ has been implemented. The function takes as input x and the image f and gives as output the result of the interpolation, by using the formulation given in equation (4).

$$F_g(x) = \sum_{i=1}^{7} g(x-i)f(i) \tag{4}$$

```
function [result] = funct_new(x,f)
    result  = 0;
    for i=1:7
        result = result + funct(x-i)*f(i);
    end
end
```

Listing 3: Matlab code for 2.c - Fg(x)

The interpolation $F_g(x)$ of the function f using $g(x)$ is illustrated in Figure 5. As it can be observed, the function is defined in every point, therefore it is continuous. Moreover, it does not have any jumps (the graphic has a smooth shape), thus it is also differentiable.

Completion after feedback From the mathematical point of view, it is known that a function is differentiable at a point a if the derivative

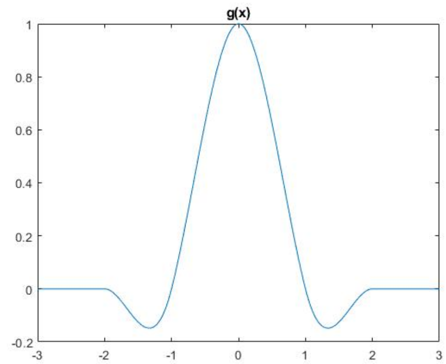

Figure 4: Function g(x)

from equation (5) exists.

$$f^{'}(0) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h} \tag{5}$$

But $F_g(x)$ is an interpolation of the function f using $g(x)$. Therefore, one could use the function $g(x)$ to prove that $F_g(x)$ is differentiable.

$$\lim_{|x| \leftarrow 1^-} g(x) = |x|^3 - 2|x|^2 + 1 = 0 \tag{6}$$

$$\lim_{|x| \to 1^+} g(x) = -|x|^3 + 5|x|^2 - 8|x| + 4 = 0 \tag{7}$$

Therefore, $\lim_{|x| \leftarrow 1^-} g(x) = \lim_{|x| \to 1^+} g(x) = 0$.

$$\lim_{|x| \leftarrow 2^-} g(x) = -|x|^3 + 5|x|^2 - 8|x| + 4 = 0 \tag{8}$$

$$\lim_{|x| \to 2^+} g(x) = 0 \tag{9}$$

Therefore, $\lim_{|x| \leftarrow 1^-} g(x) = \lim_{|x| \to 1^+} g(x) = \lim_{|x| \leftarrow 2^-} g(x) = \lim_{|x| \to 2^+} g(x) = 0$. So the derivative exists in every point, consequently the function $g(x)$ is differentiable. Thus, $F_g(x)$ is also differentiable.

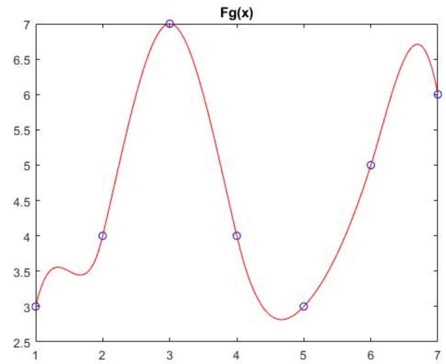

Figure 5: Function Fg(x)

3 Classification using Nearest Neighbour and Bayes theorem

3.1 Nearest Neighbours

In order to get the KNN classification model, the Matlab function *fitcknn* can be used. The function takes as input the training input features (first four measurements from each class), and the training output features (the class corresponding to the input).

Then, the predicted values can be obtained by using the Matlab function *predict*. This function takes as input the KNN model and the testing features (last three measurements from each class). Therefore, the Matlab code for this task is:

```
x_train_1 = [0.4003 0.3988 0.3998 0.3997];
x_test_1 = [0.4010 0.3995 0.3991];

x_train_2 = [0.2554 0.3139 0.2627 0.3802];
x_test_2 = [0.3287 0.3160 0.2924];

x_train_3 = [0.5632 0.7687 0.0524 0.7586];
x_test_3 = [0.4243 0.5005 0.6769];

X_train = [x_train_1 x_train_2 x_train_3]';
X_test = [x_test_1 x_test_2 x_test_3]';
y_train = [1 1 1 1 2 2 2 2 3 3 3 3]';
```

```
y_test = [1 1 1 2 2 2 3 3 3]';

mdl = fitcknn(X_train,y_train)
flwrClass = predict(mdl,X_test)
```

Listing 4: Matlab code for 3.a

The results can be observed in Table 1.

X test	0.4010	0.3995	0.3991	0.3287	0.3160	0.2924	0.4243	0.5005	0.6769
Predicted class	1	1	1	2	2	2	1	3	3
Expected class	1	1	1	2	2	2	3	3	3

Table 1: Results for the KNN classifiaction

As it can be seen from Table 1, only one measurement was miss-classified (the first testing measurement from the third class). Consequently, 8 measurements were correctly classified.

Completion after feedback An example of how to classify a point using the NN can be observed below. Let's take the first point from the first class, 0.4003. First, we compute the Euclidian distance between this point and all the testing points.

$$dist = \sqrt{|0.4003 - 0.4010|^2} = 0.0007 \tag{10}$$

$$dist = \sqrt{|0.4003 - 0.3995|^2} = 0.0008 \tag{11}$$

$$dist = \sqrt{|0.4003 - 0.3991|^2} = 0.0012 \tag{12}$$

$$dist = \sqrt{|0.4003 - 0.3287|^2} = 0.0716 \tag{13}$$

$$dist = \sqrt{|0.4003 - 0.3160|^2} = 0.0843 \tag{14}$$

$$dist = \sqrt{|0.4003 - 0.2924|^2} = 0.1079 \tag{15}$$

$$dist = \sqrt{|0.4003 - 0.4243|^2} = 0.024 \tag{16}$$

$$dist = \sqrt{|0.4003 - 0.5005|^2} = 0.1002 \tag{17}$$

$$dist = \sqrt{|0.4003 - 0.67691|^2} = 0.27661 \tag{18}$$

By comparing all the distances, it can be seen that the minimal distance is 0.0007, corresponding to the measurement 0.4010, from the first class. Thus, the point 0.4003 is classified as *class1*.

3.2 Gaussian distributions

First, Matlab can be used to get the pdf of the normal distribution with each of the three given means and deviations, evaluated at all seven samples in each class. This can be done with the following code:

```
Y11 = normpdf([x_train_1 x_test_1; x_train_2 x_test_2;
   x_train_3 x_test_3],0.4,0.01)
Y22 = normpdf([x_train_1 x_test_1; x_train_2 x_test_2;
   x_train_3 x_test_3],0.3,0.05)
Y33 = normpdf([x_train_1 x_test_1; x_train_2 x_test_2;
   x_train_3 x_test_3],0.5,0.2)
```

Listing 5: Matlab code for 3.b

After that, the results were arranged according to the three datasets. The results can be observed in Tables 2, 3, 4.

Class 1	39.8763	39.6080	39.8862	39.8763	39.6953	39.8444	39.7330
Class 2	1.0669	1.1326	1.0885	1.0928	1.0373	1.1016	1.1192
Class 3	1.7616	1.7550	1.7594	1.7590	1.7647	1.7581	1.7564

Table 2: Results for the Gaussian Distributions - First Data Set

Class 1	0.0000	0.0000	0.0000	5.6183	0.0000	0.0000	0.0000
Class 2	5.3600	7.6764	6.0408	2.2042	6.7670	7.5806	7.8872
Class 3	0.9442	1.2938	0.9867	1.6671	1.3822	1.3064	1.1639

Table 3: Results for the Gaussian Distributions - Second Data Set

Class 1	0.0000	0.0000	0.0000	0.0000	2.0829	0.0000	0.0000
Class 2	0.0000	0.0000	0.0000	0.0000	0.3630	0.0026	0.0000
Class 3	1.8976	0.8090	0.1630	0.8647	1.8568	1.9947	1.3490

Table 4: Results for the Gaussian Distributions - Third Data Set

Completion after feedback The Bayes Theorem [1] has the following equation:

$$P(y = j|x) = \frac{P(x|y = j)P(y = j)}{P(x)} \tag{19}$$

In this case, the likelihood, or the term $P(x|y = j)$ is given by the normal distribution. But it is stated that all classes are equally likely to occur, therefore, by taking this into account and the Bayes Theorem, one could say that the maximum a posteriori classifier of a point x is in fact the class of the gaussian distribution with the highest value from all gaussian distributions of the point x. So we have:

$$j = argmax_k P(y = k|x) = argmax(N(x, m_1, \sigma_1), N(x, m_2, \sigma_2), N(x, m_3, \sigma_3)) \tag{20}$$

Thus, the next step is to look in tables for the maximal values at every column. As it can be observed, for the first dataset (Table 2) all the maximal values per column are in the first row. This is good, because the first raw corresponds to the first class. So all seven values from the first dataset are correctly classified. The maximum a posteriori classification for the first data set is 1, corresponding to the maximal gaussian distribution value, 39.8862. After that, one can look for the maximal values at every column in Table 3. There is a miss-classification in the fourth column (the orange value). In this case, the maximal value is found in the first raw (corresponding to the first class), not in the second one (corresponding to the second class). All the other maximal values are in the second raw. So only six values are correctly classified for the second dataset.The maximum a posteriori classification for the second data set is 2, corresponding to the maximal gaussian distribution value, 7.8872. Finally, after inspecting the values in Table 4, a new miss-classification can be noticed. In the fifth column, the maximal value is in the first raw (corresponding to the first class), and not in the third raw (corresponding to the third class). All the other maximal column values are in the third raw. Consequently, six values were correctly classified for the third dataset. The maximum a posteriori classification for the first data set is 3, corresponding to the maximal gaussian distribution value, 1.9947.

4 Classification

Let $a = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $b = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $c = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$, and $x = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$. Moreover, let $P(0|1) = P(1|0) = \epsilon$ and $P(0|0) = P(1|1) = 1 - \epsilon$. By using the Bayes theorem, we get the following equations:

$$P(a|x) = \frac{P(x|a)P(a)}{P(x)} \tag{21}$$

$$P(b|x) = \frac{P(x|b)P(b)}{P(x)} \tag{22}$$

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{23}$$

By replacing the names of the matrices with their value, other equations are obtained:

$$P(x|a) = (1-\epsilon)^3\epsilon \tag{24}$$

$$P(x|b) = (1-\epsilon)\epsilon^3 \tag{25}$$

$$P(x|c) = (1-\epsilon)^2\epsilon^2 \tag{26}$$

Moreover, it is known that

$$\frac{P(x|a)P(a)}{P(x)} + \frac{P(x|b)P(b)}{P(x)} + \frac{P(x|c)P(c)}{P(x)} = 1 \tag{27}$$

This can be rewritten as:

$$P(x|a)P(a) + P(x|b)P(b) + P(x|c)P(c) = P(x) \tag{28}$$

Therefore, for $\epsilon = 0.05$, the following results are obtained:

$$P(a|x) = 0.945 \tag{29}$$

$$P(b|x) = 0.0052 \tag{30}$$

$$P(c|x) = 0.0497 \tag{31}$$

Consequently, by analysing the values above, one could say that $P(a|x)$ has the maximal value $\epsilon = 0.05$.

However, for $\epsilon = 0.5$, the following results are obtained:

$$P(x) = 0.0625 \tag{32}$$

$$P(a|x) = 0.25 \tag{33}$$

$$P(b|x) = 0.5 \tag{34}$$

$$P(c|x) = 0.25 \tag{35}$$

Consequently, by analysing the values above, one could say that $P(b|x)$ has the maximal value for $\epsilon = 0.5$.

5 Classification

Let $I_1 = \begin{pmatrix} b & w & w & w \\ b & w & w & w \\ b & w & w & w \\ b & w & w & w \end{pmatrix}$, $I_2 = \begin{pmatrix} w & b & w & w \\ w & b & w & w \\ w & b & w & w \\ w & b & w & w \end{pmatrix}$, $I_3 = \begin{pmatrix} w & w & b & w \\ w & w & b & w \\ w & w & b & w \\ w & w & b & w \end{pmatrix}$, $I_4 = \begin{pmatrix} w & w & w & b \\ w & w & w & b \\ w & w & w & b \\ w & w & w & b \end{pmatrix}$, and $x = \begin{pmatrix} b & w & w & w \\ w & b & w & w \\ w & w & b & w \\ w & b & w & w \end{pmatrix}$ where b represents a black coloured pixel and w represents a white coloured pixel. Moreover, let $P(w|b) = P(b|w) = \epsilon$, $P(b|b) = P(w|w) = 1 - \epsilon$, $P(I_1) = P(I_4) = 0.3$, and $P(I_2) = P(I_3) = 0.2$. By using the Bayes theorem, we get the following equations:

$$P(I_1|x) = \frac{P(x|I_1)P(I_1)}{P(x)} \tag{36}$$

$$P(I_2|x) = \frac{P(x|I_2)P(I_2)}{P(x)} \tag{37}$$

$$P(I_3|x) = \frac{P(x|I_3)P(I_3)}{P(x)} \tag{38}$$

$$P(I_4|x) = \frac{P(x|I_4)P(I_4)}{P(x)} \tag{39}$$

By replacing the names of the matrices with their value, other equations are obtained:

$$P(x|I_1) = (1-\epsilon)^{10}\epsilon^6 \tag{40}$$

$$P(x|I_2) = (1-\epsilon)^{12}\epsilon^4 \tag{41}$$

$$P(x|I_3) = (1-\epsilon)^{10}\epsilon^6 \tag{42}$$

$$P(x|I_4) = (1-\epsilon)^{8}\epsilon^8 \tag{43}$$

Moreover, it is known that

$$\frac{P(x|I_1)P(I_1)}{P(x)} + \frac{P(x|I_2)P(I_2)}{P(x)} + \frac{P(x|I_3)P(I_3)}{P(x)} + \frac{P(x|I_4)P(I_4)}{P(x)} = 1 \tag{44}$$

This can be rewritten as:

$$P(x|I_1)P(I_1) + P(x|I_2)P(I_2) + P(x|I_3)P(I_3) + P(x|I_4)P(I_4) = P(x) \tag{45}$$

Therefore, for $\epsilon = 0.2$, the following results are obtained:

$$P(x) = 2.55e^{-5} \tag{46}$$

$$P(I_1|x) = 0.0807 \tag{47}$$

$$P(I_2|x) = 0.8605 \tag{48}$$

$$P(I_3|x) = 0.0538 \tag{49}$$

$$P(I_4|x) = 0.0050 \tag{50}$$

Consequently, by analysing the values above, one could say that I_2 is the most probable image for $\epsilon = 0.2$.

6 Classification

Let $w_1 = \begin{pmatrix} b & b & w \\ b & w & b \\ b & b & w \\ b & w & b \\ b & b & b \end{pmatrix}$, $w_2 = \begin{pmatrix} w & b & w \\ b & w & b \\ b & w & b \\ b & w & b \\ w & b & w \end{pmatrix}$, $w_3 = \begin{pmatrix} w & b & w \\ b & w & b \\ w & b & w \\ b & w & b \\ w & b & w \end{pmatrix}$, and $x = \begin{pmatrix} w & w & w \\ b & w & w \\ w & b & w \\ w & w & b \\ b & b & w \end{pmatrix}$, where b represents a black coloured pixel and w represents a white coloured pixel. Moreover, let $P(b|w) = 0.35$, $P(w|w) = 1-0.35 = 0.65$, $P(w|b) = 0.25$, $P(b|b) = 1 - 0.25 = 0.75$, $P(w_1) = 0.25$, $P(w_2) = 0.4$, and $P(w_3) = 0.35$. By using the Bayes theorem, we get the following equations:

$$P(w_1|x) = \frac{P(x|w_1)P(w_1)}{P(x)} \tag{51}$$

$$P(w_2|x) = \frac{P(x|w_2)P(w_2)}{P(x)} \tag{52}$$

$$P(w_3|x) = \frac{P(x|w_3)P(w_3)}{P(x)} \tag{53}$$

By replacing the names of the matrices with their value, other equations are obtained:

$$P(x|w_1) = P(w|b)^5 P(b|b)^5 P(w|w)^5 \tag{54}$$

$$P(x|w_2) = P(b|w)^2 P(w|b)^5 P(b|b)^3 P(w|w)^5 \tag{55}$$

$$P(x|w_3) = P(b|w) P(w|b)^3 P(b|b)^4 P(w|w)^7 \tag{56}$$

Moreover, it is known that

$$\frac{P(x|w_1)P(w_1)}{P(x)} + \frac{P(x|w_2)P(w_2)}{P(x)} + \frac{P(x|w_3)P(w_3)}{P(x)} = 1 \tag{57}$$

This can be rewritten as:

$$P(x|w_1)P(w_1) + P(x|w_2)P(w_2) + P(x|w_3)P(w_3) = P(x) \tag{58}$$

Therefore, for the given probabilities, the following results are obtained:

$$P(x) = 3.875e^{-5} \tag{59}$$

$$P(w_1|x) = 0.1735 \tag{60}$$

$$P(w_2|x) = 0.0604 \tag{61}$$

$$P(w_3|x) = 0.7661 \tag{62}$$

Consequently, by analysing the values above, one could say that w_3 is the most probable image.

7 The OCR system - part 2 - Feature extraction

First of all, some properties of the image regions given by the Matlab function *regionprops* have been taken into account. The digits have different sizes and different number of white pixels. Therefore, it seemed a good idea to include the area and the perimeter in the feature vector.

Also, some digits have holes inside (e.g. 0, 6, 8, 9). In this context, the Euler number might be a good property, as it is defined as the difference between the total number of objects in the image and the total number of holes in the objects [2]. Therefore, negative and 0 values of the Euler Number should be expected for the digits with holes.

The centroid may also be useful, as it is the average of the pixels forming a shape. So, depending on the shape of a digit, the centroid has differerent coordinates.

In addition, the Matlab function *imdilate* does a dilation operation on the image. In this way, a bigger digit should be obtained, with filled holes. Consequently, another way to differentiate between the digits with holes and the digits without holes may be to compute the difference between the area of the dilated digit and the area with the original digit. In this way, the difference would be smaller for the digits without holes and bigger for the digit with holes.

Lastly, the number of edges and corners may be slightly different from one digit to another. The edges of an image can be obtained easily, by using the Matlab function *edge*. Then, these can be counted by using the function *nnz*, which counts the number of non-zero elements of a matrix. As far as the corners are concerned, the corner points can be obtained by using the Matlab function *corner* and the number of corners is the length of the array with the corner points.

The code for the feature vector can be observed bellow.

```
function features = segment2features(I)
% features = segment2features(I)
    CC =  bwconncomp(I,8);
    A = regionprops(CC,'Area');
    P = regionprops(CC,'Perimeter');
    EN = regionprops(CC,'EulerNumber');
    C = regionprops(CC, 'Centroid');
    edges = edge(I,'canny');
    countEdges = nnz(edges); %number of nonzero matrix elements
    se = strel('sphere',5);
    I_filled = imdilate(I, se);
    CC_filled =  bwconncomp(I_filled,8);
    A_filled = regionprops(CC_filled,'Area');
    diff_filled = A_filled.Area - A.Area;
    corneres = corner(I);
    noCorners = length(corneres);
    features = [A.Area P.Perimeter EN.EulerNumber C.Centroid ...
```

```
        diff_filled countEdges noCorners]';
```

Listing 6: Matlab code for 7

The clustered digits can be observed in Figure 6

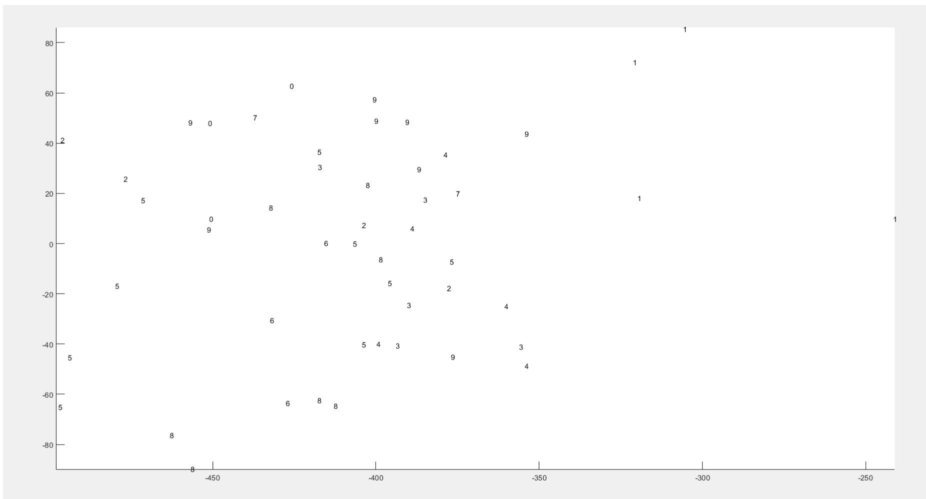

Figure 6: Clustered digits

In addition, the feature vector for some input data (different characters and different images of the same character) can be observed in the results below.

```
>> inl2_test_and_benchmark
Studying the character 0
There are 3 examples in the database.
The feature vectors for these are:

ans =

  100.0000  137.0000  100.0000
   50.3570   52.8220   48.7690
         0         0         0
   69.4000  125.9927  125.6500
   14.8500   16.3139   14.6500
  423.0000  399.0000  387.0000
   99.0000   98.0000   90.0000
    6.0000    9.0000    6.0000

Studying the character 1
```

```
There are 4 examples in the database.
The feature vectors for these are:

ans =

   59.0000   89.0000   55.0000   83.0000
   40.5820   50.2990   31.4680   48.9760
    1.0000    1.0000    1.0000    1.0000
  127.5254   68.1236   41.1455  124.0361
   15.4915   16.3146   13.2182   16.5060
  278.0000  291.0000  229.0000  285.0000
   47.0000   53.0000   32.0000   50.0000
    4.0000    4.0000    5.0000    5.0000

Studying the character 2
There are 4 examples in the database.
The feature vectors for these are:

ans =

  119.0000  146.0000  112.0000  114.0000
   87.5900   88.2020   69.2550   62.8120
         0    1.0000    1.0000         0
   96.8824  124.9521   69.7857   43.4561
   15.8739   14.2534   15.7411   16.8772
  434.0000  441.0000  369.0000  343.0000
   97.0000   95.0000   72.0000   83.0000
    8.0000   10.0000    7.0000    4.0000

Studying the character 3
There are 5 examples in the database.
The feature vectors for these are:

ans =

  102.0000  128.0000  120.0000  107.0000   88.0000
   89.2440   66.6250   71.4460   60.9930   66.1540
    1.0000         0    1.0000    1.0000    1.0000
   15.8824   41.3125   98.6167   13.4486   69.9205
   16.2451   16.0469   15.4917   14.1495   14.4432
  365.0000  354.0000  375.0000  331.0000  359.0000
   84.0000   71.0000   74.0000   65.0000   64.0000
    8.0000    9.0000    8.0000    6.0000    8.0000

Studying the character 4
```

```
There are 5 examples in the database.
The feature vectors for these are:

ans =

  132.0000  123.0000  110.0000  128.0000  167.0000
   50.5890   66.7080   63.2090   49.1830   55.1790
   -1.0000         0    1.0000   -1.0000         0
   41.0606   72.3415   98.4091   13.0938   40.4371
   16.0455   17.0488   17.4727   15.6172   15.9581
  322.0000  349.0000  338.0000  323.0000  351.0000
   72.0000   70.0000   69.0000   68.0000   79.0000
    8.0000    7.0000    8.0000    7.0000    8.0000

Studying the character 5
There are 9 examples in the database.
The feature vectors for these are:

ans =

  104.0000   98.0000  141.0000  176.0000  221.0000  132.0000
     83.0000  168.0000  103.0000
   73.8530   76.2440   89.7310  100.2260  112.7650   71.5290
     67.8240   92.6260   73.3810
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
      1.0000    1.0000    1.0000
   42.3077   13.8980   96.3546   44.4659   42.9864   70.5303
     41.7711   70.8036   98.8155
   15.6346   16.6633   13.7660   16.3807   14.4118   14.5455
     16.6988   16.3036   12.7282
  366.0000  381.0000  421.0000  439.0000  423.0000  363.0000
    355.0000  424.0000  380.0000
   79.0000   83.0000   92.0000  103.0000  109.0000   79.0000
     72.0000   94.0000   78.0000
    7.0000    7.0000   10.0000   12.0000   16.0000    6.0000
      5.0000   15.0000    8.0000

Studying the character 6
There are 3 examples in the database.
The feature vectors for these are:

ans =

  126.0000  138.0000  159.0000
   66.1370   69.2960   68.4440
```

```
         0         0         0
   69.5238   41.7899   14.3019
   15.5556   16.3043   16.8113
  373.0000  391.0000  384.0000
   93.0000   97.0000   94.0000
    6.0000    6.0000   13.0000

Studying the character 7
There are 2 examples in the database.
The feature vectors for these are:

ans =

   83.0000  136.0000
   59.7440   72.0250
    1.0000         0
   70.2410  127.6618
   10.8313   11.6912
  350.0000  384.0000
   68.0000   82.0000
    5.0000    8.0000

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

ans =

  159.0000  164.0000  226.0000  149.0000  163.0000  193.0000
    136.0000
   59.7690   56.7500   62.7460   53.9550   59.8270   63.7880
     48.9680
   -1.0000   -1.0000   -1.0000   -2.0000   -1.0000   -1.0000
     -1.0000
   13.6541   13.8902   14.2611   69.7315  100.0613   15.5751
     96.4926
   13.1195   13.1402   15.1062   13.2752   16.8650   16.3264
     13.9853
  378.0000  369.0000  391.0000  351.0000  373.0000  409.0000
    355.0000
   81.0000   90.0000  106.0000   80.0000   96.0000  108.0000
     84.0000
    8.0000   12.0000   16.0000   10.0000   10.0000   12.0000
      8.0000

```

```
Studying the character 9
There are 8 examples in the database.
The feature vectors for these are:

ans =

  131.0000  130.0000   95.0000  142.0000  145.0000  116.0000
    182.0000  122.0000
   63.8710   52.8960   48.9100   61.7370   61.1330   56.9660
     64.5160   53.1200
         0         0         0   -1.0000         0         0
           0         0
  124.5802  100.7000  100.4737  126.8732  126.9241  125.9052
     98.8407   16.4262
   14.4962   15.0000   15.5789   14.6479   14.6000   12.6379
     14.1044   15.4590
  406.0000  339.0000  317.0000  344.0000  333.0000  353.0000
    387.0000  346.0000
  100.0000   80.0000   73.0000   75.0000   72.0000   84.0000
     96.0000   82.0000
    7.0000    8.0000    4.0000    8.0000    7.0000    5.0000
     13.0000    6.0000
```
Listing 7: Results

References

[1] M. Oskarsson, Image Analysis Course (2021), Lund University.
[2] Mathworks, https://www.mathworks.com/help/images/ref/bweuler.html