

Image Analysis Assignment 2

Olof Harrysson

1.

To figure this out I flip the filter in both the x and y direction.

f1 = A. The filter detects when the image goes from white to black in the direction left to right.

f2 = D. Similarly like f1 the filter detects edges in the y-direction.

f3 = C. Detects the border outside of a black object.

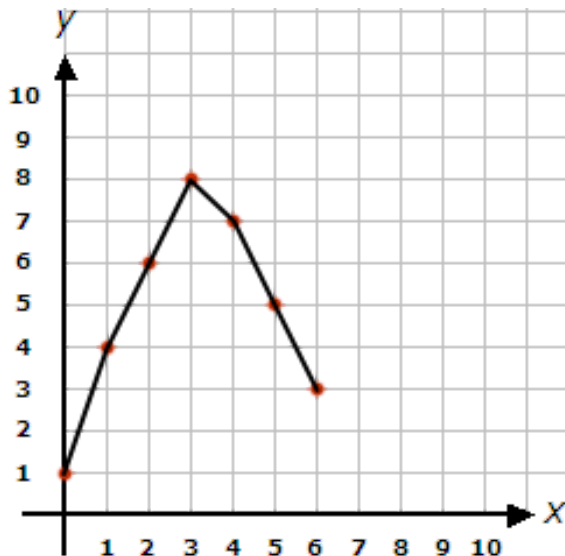
f4 = F. The filter blurs the image which is hard to see for this size.

f5 = E. Detects bottom right edges.

f6 = B. Detects bottom left edges.

2.

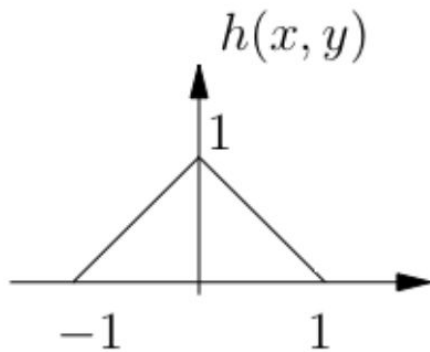
a) Linear interpolation connects sequential points by using straight lines.



b) $g(x) = 0$ when $x < -1$

$g(x) = 1 - |x|$ when $-1 \leq x \leq 1$

$g(x) = 0$ when $1 < x$



```
0.5 interpolates to 2.5
1.5 interpolates to 5.0
2.5 interpolates to 7.0
3.5 interpolates to 7.5
4.5 interpolates to 6.0
5.5 interpolates to 4.0
```

```
Derivate in 0.5 is 3.0
Derivate in 1.5 is 2.0
Derivate in 2.5 is 2.0
Derivate in 3.5 is -1.0
Derivate in 4.5 is -2.0
Derivate in 5.5 is -2.0
```

```
f * w = [ 3  2  2 -1 -2 -2]
```

c)

The derivative vector is the same as $f * w$.

$$d) (1) \frac{\phi F(k)}{\phi x} = \sum_i \frac{\phi g(x-i)}{\phi x} \times f(k), \quad \frac{\phi g(x-i)}{\phi x} =$$

0 when $x < -1$

1 when $-1 < x < 0$

-1 when $0 < x < 1$

0 when $1 < x$

$$(2) \sum_i w(x-i) \times f(i)$$

If you put equation (1) = (2) you can cross out the $f(i)$ and the sum and what's left is $\frac{\phi g(j)}{\phi x} = w(j)$

e) Using the result from the last problem we can now construct a $g'(x)$ = such that

$1/2$ when $x < 2$

0 when $2 < x < 4$

$-1/2$ when $4 < x$

From there we construct a $g(x)$ =

$x/2$ when $x < 2$

0 when $2 < x < 4$

$3 - x/2$ when $4 < x$

3.

a) The items in the test data is compared against the training data by looking at the distance between each test and train pair. $err = |test\ item - train\ item|$

The test item is then classified as the class where the error is the smallest.

```
0.4243 is wrongly classified as Class 1
Number of missclassified items are 1
Olofs-MacBook-Pro:2 olof$
```

b) I run each data point through three normal distribution models, one for each class. The data point is classified to the class that corresponds to the highest probability.

```
0.3802 is wrongly classified as Class 1
0.4243 is wrongly classified as Class 1
Number of missclassified items are 2
Olofs-MacBook-Pro:2 olof$
```

4.

a)

$$P(X|Y = 1) = 0.1^1 * 0.9^3 = 0.0729$$

$$P(X|Y = 2) = 0.1^3 * 0.9^1 \approx 9 * 10^{-4}$$

$$P(X|Y = 3) = 0.1^2 * 0.9^2 \approx 8.1 * 10^{-6}$$

$$P(1) = 0.0729 / (0.0729 + 9 * 10^{-4} + 8.1 * 10^{-6}) \approx 0.988$$

$$P(2) = 9 * 10^{-4} / (0.0729 + 9 * 10^{-4} + 8.1 * 10^{-6}) \approx 0.012$$

$$P(3) = 8.1 * 10^{-6} / (0.0729 + 9 * 10^{-4} + 8.1 * 10^{-6}) \approx 0.000109$$

5.

The probability of the image looking the way it does given that the line was in column i is the probability of the correct pixels multiplied with the incorrect ones. For example, if the line would be in column 1 the top row of pixels would all be correct. The second row would have two incorrect pixels as the two most leftward pixels are swapped.

$$P(X|Y = 1) = 0.8^{10} * 0.2^6 \approx 6.872 * 10^{-6}$$

$$P(X|Y = 2) = 0.8^{12} * 0.2^4 \approx 1.099 * 10^{-4}$$

$$P(X|Y = 3) = 0.8^{10} * 0.2^6 \approx 6.872 * 10^{-6}$$

$$P(X|Y = 4) = 0.8^8 * 0.2^8 \approx 4.295 * 10^{-7}$$

Multiply these with the prior probability of the line being in the column.

$$w_1 = 6.872 * 10^{-6} * 0.3 = 2.062 * 10^{-6}$$

$$w_2 = 1.099 * 10^{-4} * 0.2 = 2.198 * 10^{-6}$$

$$w_3 = 6.872 * 10^{-6} * 0.2 = 1.3744 * 10^{-6}$$

$$w_4 = 4.295 * 10^{-7} * 0.3 = 1.289 * 10^{-6}$$

It turns out that the column two has the highest number and therefor probability. To calculate the probability of column 2, we take w_2 and divide with the sum of all w .

The probability of the line being in column 2 is 86%

6.

Similarly to the last exercise we want to figure out what the probability of the image looking the way it does given the original image.

$$P(X|Y = B) = 0.2^5 * 0.3^0 * 0.7^5 * 0.8^5 \approx 1.762 * 10^{-5}$$

$$P(X|Y = 0) = 0.2^5 * 0.3^2 * 0.7^5 * 0.8^3 \approx 2.478 * 10^{-6}$$

$$P(X|Y = 8) = 0.2^3 * 0.3^1 * 0.7^7 * 0.8^4 \approx 8.096 * 10^{-5}$$

$$w_b = 1.762 * 10^{-5} * 0.3 = 5.286 * 10^{-6}$$

$$w_0 = 2.478 * 10^{-6} * 0.4 = 9.912 * 10^{-7}$$

$$w_8 = 8.096 * 10^{-5} * 0.3 = 2.429 * 10^{-5}$$

$$P(B) = 5.286 * 10^{-6} / (5.286 * 10^{-6} + 9.912 * 10^{-7} + 2.429 * 10^{-5}) \approx 0.173$$

$$P(0) = 9.912 * 10^{-7} / (5.286 * 10^{-6} + 9.912 * 10^{-7} + 2.429 * 10^{-5}) \approx 0.032$$

$$P(8) = 2.429 * 10^{-5} / (5.286 * 10^{-6} + 9.912 * 10^{-7} + 2.429 * 10^{-5}) \approx 0.794$$

7.

```
function features = segment2features(input_img)
% features = segment2features(I)
features = zeros(6,1);

% Number of white pixels in the image
% I imagine this could separate for example I and B as B has more
% white pixels.
brightness = sum(sum(input_img));
features(1) = brightness / 10;

% Number of white pixels in the whitest column
% Could detect if the letter contains a long vertical line.
% Compare C vs P
brightest_col = max(sum(input_img));
features(2) = brightest_col;

% Number of white pixels in the whitest row
% Similar to the previous feature but horizontal.
% Compare T vs J
brightest_row = max(sum(input_img, 2));
features(3) = brightest_row;

% Number of segments in the inverted image
% Some letters such as A or B encloses part of the image within
% themselves that isn't a part of letter pixels. By inverting the
% image and running the bwlabel, we can detect how many such
% segments there are. J has 0, A has 1, B has two enclosed segments.
compl_img = imcomplement(input_img);
[labels, number_seg] = bwlabel(compl_img, 8);
features(4) = number_seg * 10;

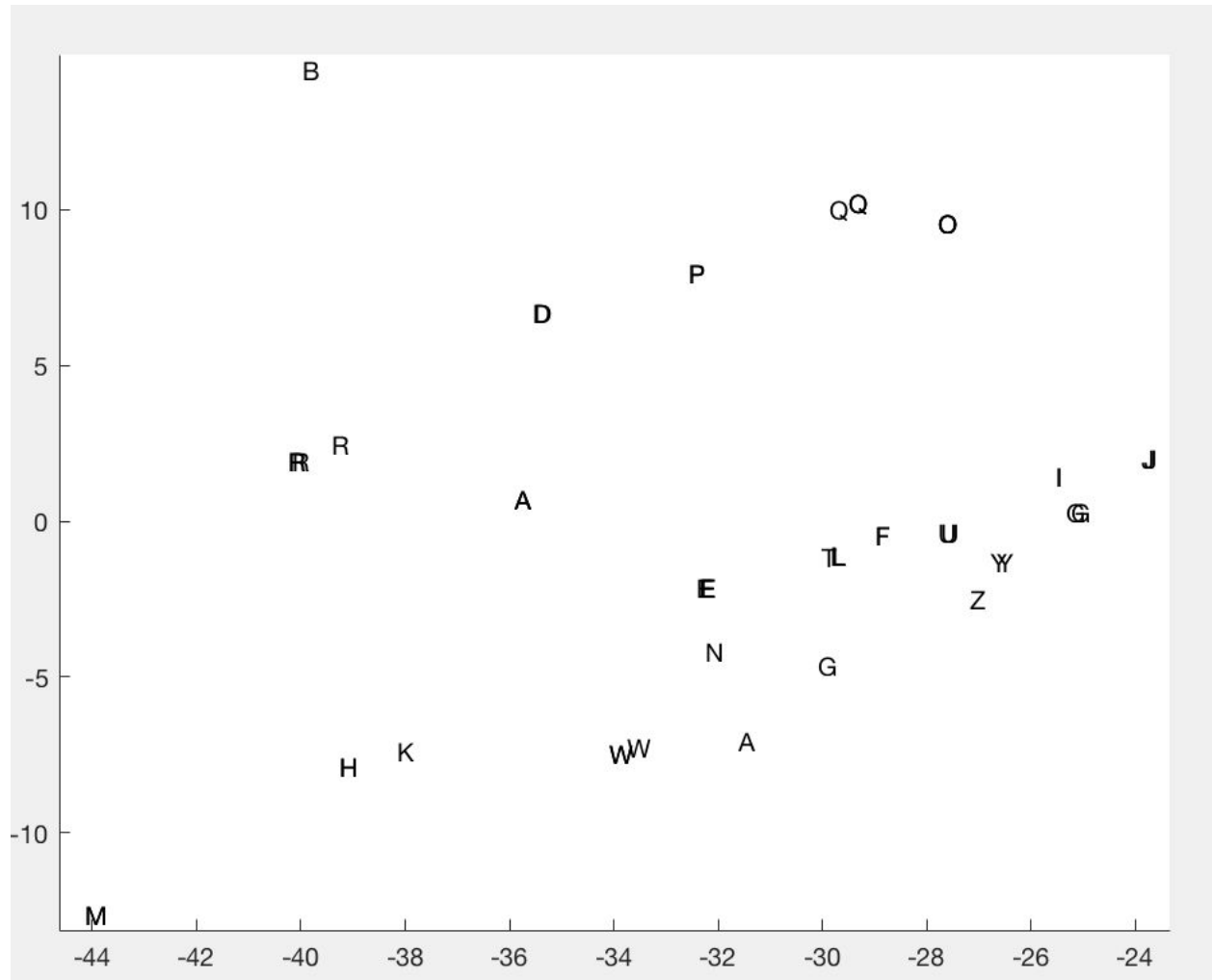
% Number of "feet"
% Taking the first row that contains a white pixel, starting from
% the bottom, and check how many segments that is.
% Compare A vs B

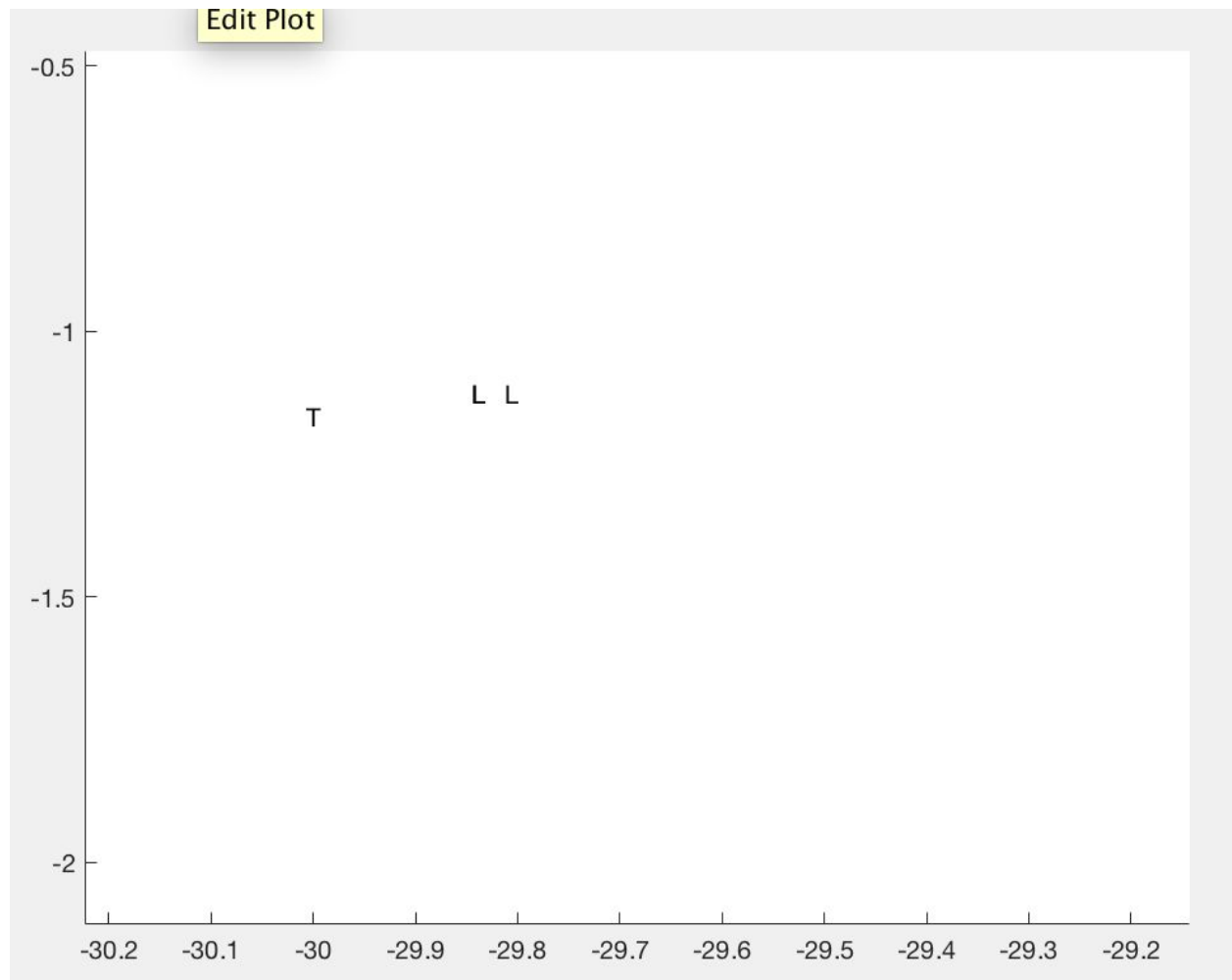
% Feet width
% Similar to the previous feature I check the bottom row that contains
% a white pixel. I then take the sum of that row.
% Compare L vs C
nbr_feet = 0;
feet_length = 0;
for index_seg = size(input_img,1): -1 : 1
    if sum(input_img(index_seg,:)) ~= 0
        [segment_label, nbr_segments] = bwlabel(input_img(index_seg,:), 4);
        nbr_feet = nbr_segments;
        feet_length = sum(input_img(index_seg,:));
    end
end
```

```

        break;
    end
end
features(5) = nbr_feet * 10;
features(6) = feet_length;

```





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

ans =

9.3000	9.4000	9.4000	9.4000
10.0000	10.0000	10.0000	10.0000
14.0000	14.0000	14.0000	14.0000
10.0000	20.0000	20.0000	20.0000
20.0000	20.0000	20.0000	20.0000
14.0000	14.0000	14.0000	14.0000

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

ans =

10.6000	10.6000	10.8000	10.7000
17.0000	17.0000	17.0000	17.0000
16.0000	16.0000	16.0000	16.0000
10.0000	10.0000	10.0000	10.0000
10.0000	10.0000	10.0000	10.0000
16.0000	16.0000	16.0000	16.0000

<p>Studying the character H There are 2 examples in the database. The feature vectors for these are:</p> <p>ans =</p> <pre> 13.4000 13.4000 17.0000 17.0000 18.0000 18.0000 10.0000 10.0000 20.0000 20.0000 18.0000 18.0000 </pre>	<p>Studying the character L There are 3 examples in the database. The feature vectors for these are:</p> <p>ans =</p> <pre> 7.7000 7.7000 7.6000 17.0000 17.0000 17.0000 14.0000 14.0000 14.0000 10.0000 10.0000 10.0000 10.0000 10.0000 10.0000 14.0000 14.0000 14.0000 </pre>
<p>Studying the character O There are 3 examples in the database. The feature vectors for these are:</p> <p>ans =</p> <pre> 10.4000 10.4000 10.4000 13.0000 13.0000 13.0000 7.0000 7.0000 7.0000 20.0000 20.0000 20.0000 10.0000 10.0000 10.0000 7.0000 7.0000 7.0000 </pre>	<p>Studying the character Q There are 3 examples in the database. The feature vectors for these are:</p> <p>ans =</p> <pre> 13.0000 13.1000 13.0000 16.0000 16.0000 16.0000 9.0000 9.0000 9.0000 20.0000 20.0000 20.0000 10.0000 10.0000 10.0000 3.0000 4.0000 3.0000 </pre>
<p>Studying the character W There are 3 examples in the database. The feature vectors for these are:</p> <p>ans =</p> <pre> 15.4000 15.4000 15.4000 11.0000 11.0000 11.0000 22.0000 22.0000 22.0000 10.0000 10.0000 10.0000 20.0000 20.0000 20.0000 4.0000 3.0000 4.0000 </pre>	<p>Studying the character T There are 1 examples in the database. The feature vectors for these are:</p> <p>ans =</p> <pre> 8.7000 17.0000 16.0000 10.0000 10.0000 11.0000 </pre>

The chosen features seem to do a good job of separating many of the characters. Some of the letters such as T & L aren't separated nicely. This surprises me as I thought the feet length feature would take care of that. Comparing those feature vectors I can see that the feet length for T=11 and L=14. The feature for H=18 which tells me that this isn't working as intended.

I'm guessing that looking at the bottom row which contain a white pixel isn't robust enough. If the letter is tilted slightly you can get big differences between two "L"s. It might be better to look at the bottom 4 rows and take an average.

I also tried convolving the image but the conv2 function gave me errors that I was unable to fix