# UNIVERSITY OF OSLO

## Faculty of Mathematics and Natural Sciences

| | |
|---|---|
| **Exam:** | **INF 4300 – Digital image analysis** |
| **Date:** | **Friday December 12, 2008** |
| **Exam hours:** | **14.30-17.30** |
| **Number of pages:** | **7 pages plus 1 page enclosure** |
| **Enclosures:** | **1 sheet containing plots** |
| **Allowed aid:** | **Calculator** |

- Read the entire exercise text before you start solving the exercises. Please check that the exam paper is complete. If you lack information in the exam text or think that some information is missing, you may make your own assumptions, as long as they are not contradictory to the "spirit" of the exercise. In such a case, you should make it clear what assumptions you have made.

- Please note that all parts of the exercises have equal weight. You should spend your time in such a manner that you get to answer all exercises shortly. If you get stuck on one question, move on to the next question.

- Some of the questions are based on printed figures or plots included in the exam text. An extra copy of this sheet is included at the end of the exam text. Please draw your solution on this sheet, mark it with your candidate number and include it in your solution.

- Your answers should be **short**, typically 1-3 sentences or a sketch should be sufficient.

*Good luck!!*

## Exercise 1. Hough transform

a) What kind of preprocessing is normally done before applying the Hough transform?

At least edge detection and thresholding, can also be combined with edge- or line enhancement and noise filtering.

b) The (r,θ)-representation is given by $r = x\cos(\theta) + y\sin(\theta)$. Let N be the number of foreground pixels in the preprocessed input image. Justify/explain that the number of required operations, thus processing time, depends on N.

Hough transform has a loop over all foreground pixels, for each foreground pixel a new loop over all θ values and their corresponding r is done. Thus the processing time depends on N.

c) Explain briefly how Random Hough transform works.

Random Hough does not fill in a full line for each foreground pixel, but selects randomly two foreground pixels and increments only r, θ for this line. This is computationally much faster.

## Exercise 2. Morphology

a) How can edge detection in a binary image be computed using morphology? Suggest one possible method.

There are many possibilities here, so many answers are correct. E.g. f-(erosion(f) gives the edge, another possibility would be dilation(f)-f (outside borders).

b) You are given the image

$$\begin{bmatrix} 19 & 20 & 20 & 19 & 20 \\ 17 & 16 & 21 & 5 & 5 \\ 21 & 14 & 18 & 7 & 13 \\ 20 & 19 & 18 & 17 & 16 \\ 20 & 19 & 18 & 17 & 16 \end{bmatrix}$$

And a flat, plus-shaped structuring element: $\begin{matrix} & 1 & \\ 1 & 1 & 1 \\ & 1 & \end{matrix}$

Compute the output image after applying grey-level erosion of the image with the structuring element. Assume that the value in the output image should be 0 in positions where the structure element is partly outside the image.

Grey-level erosion corresponds to finding the local minimum for pixels covered by the structuring element.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 14 & 5 & 5 & 0 \\ 0 & 14 & 5 & 5 & 0 \\ 0 & 14 & 17 & 7 & 10 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

## Exercise 3. Feature extraction

a) Assume that we have a binary image f(x,y). A general discrete moment is defined as

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y)$$

How do you compute the center of mass $\bar{x}$ and $\bar{y}$ from this?

X_mean = m10/m00
Y_mean = m01/m00

b) Central moments are defined as

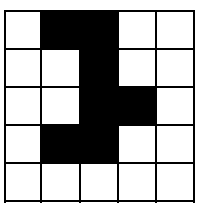
$$\mu_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q f(x,y)$$

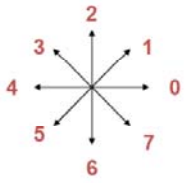
Show that $\mu_{10}=0$

Note that m00=sum(x,y)f(x,y) and this is the are or the number of object pixels in a binary image. Also note the answer for a), that X_mean = m10/m00:

$$\mu_{10} = \sum_x \sum_y (x-\bar{x}) f(x,y) = \sum_x \sum_y x f(x,y) - \sum_x \sum_y \bar{x} f(x,y) = \bar{x} m_{00} - \bar{x} m_{00} = 0$$

c) You are given the object below, where the object pixels are black. Compute the chain code for this object. Use the code table given below, and start with the top left object pixel. Search in a clockwise direction.






Start point (if upper left has coordinates 1,1): 2,1:
Chain code: 06754123

d) In general, the chain code will depend on the starting point. How can you make the code independent of the starting point?

The solution mentioned in the lecture notes is to rearrange the digits so the the corresponding number is of minimum magnitude. Comment: In the example above the regular solution is actually the number with minimum magnitude.

## Exercise 4. Texture

When measuring texture in an image, one must choose the *order* of the texture descriptor.

a) Higher order texture descriptors are more complex to calculate, but still highly useful. Explain briefly the reasons motivating the use of higher order descriptors.

Gray level co-occurrence matrices is a way of measuring texture.

b) Describe in short how to calculate a GLCM for an entire image. Explain the necessary concepts, using a sketch if you find it useful.

c) Does the number of graylevels in the image influence this measure? If so, how?

d) Does GLCM represent a rotation-invariant texture measure? If not, describe how you would handle that.

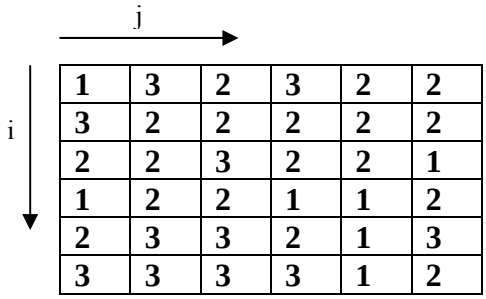
e) From the GLCM, many scalar texture measures can be derived. In the following two of these measures are given. Explain what type of characteristics of texture they measure.

$$\sum_{i=0}^{G}\sum_{j=0}^{G} P(i,j)\log(P(i,j))$$

$$\sum_{i=0}^{G}\sum_{j=0}^{G} (i-\mu)^2 P(i,j)$$

f)

For the following texture a coocurrence matrix is calculated (ignore problems with the edge pixels, i.e., do not evaluate coocurrences across the edges)




The coocurrence matrix given ***h=(i,j)*** is

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | **1** | **2** | **2** |
| 2 | **3** | **8** | **3** |
| 3 | **1** | **3** | **2** |

What is ***h***?

1. (0,1)
2. (1,0)
3. (1,1)
4. (1,-1)
5. None of the above

## Exercise 5. Classification using Bayes rule

Remember that, when assuming a 2D Gaussian distribution, the point probability of a point $x=[x_1, x_2]^T$ can be written on the form

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \frac{1}{(2\pi)^{1/2} \, |\Sigma|^{1/2}} \exp\left( -\frac{1}{2}\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}\right)^T \Sigma^{-1} \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}\right)\right)$$

where the probability distribution function takes as parameters the vector $\mu=[\mu_1, \mu_2]^T$ and a matrix $\sum$. Classification can be done by assigning a point to the class having the highest probability (i.e., the highest function *f(x)*). Any function based on a monotonous transform from the probability will give the same classification result, and is called discriminant function. By taking the logarithm of the above expression, a discriminant function can be created.

Assume two normally distributed classes with parameters

$$\mu^1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \Sigma^1 = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$$
$$\mu^2 = \begin{bmatrix} 4 \\ 2 \end{bmatrix}, \Sigma^2 = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$$

The a priori probabilies for both classes are equal.

a) Sketch the class means in a plot

b) Sketch the covariance matrices in the same plot

c) You are given the data points listed below. Insert the points in your plot. Classify each point according to the classifier specified above.

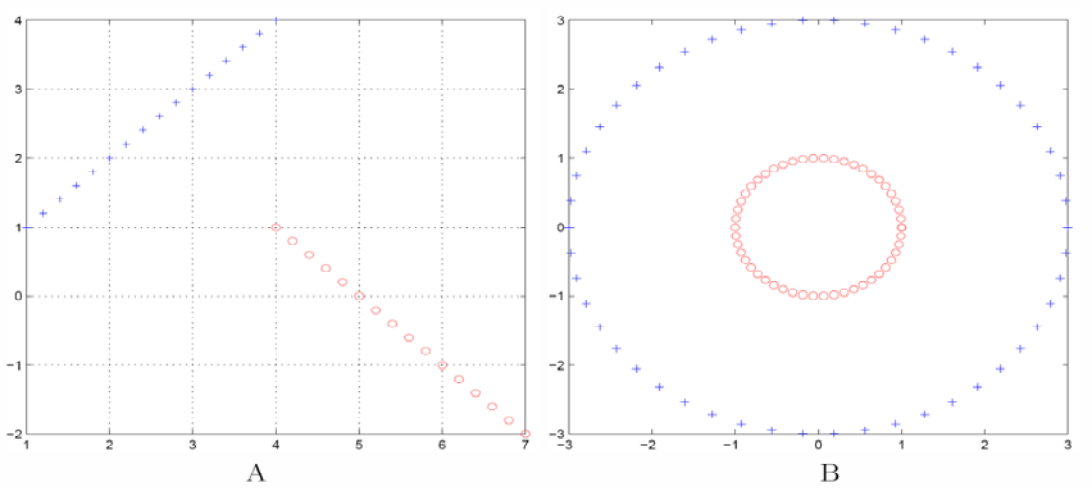
$$\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -2 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 \\ 3 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 \\ 7 \end{pmatrix}$$

d) Calculate the decision boundary.

e) Sketch the decision boundary in the plot.

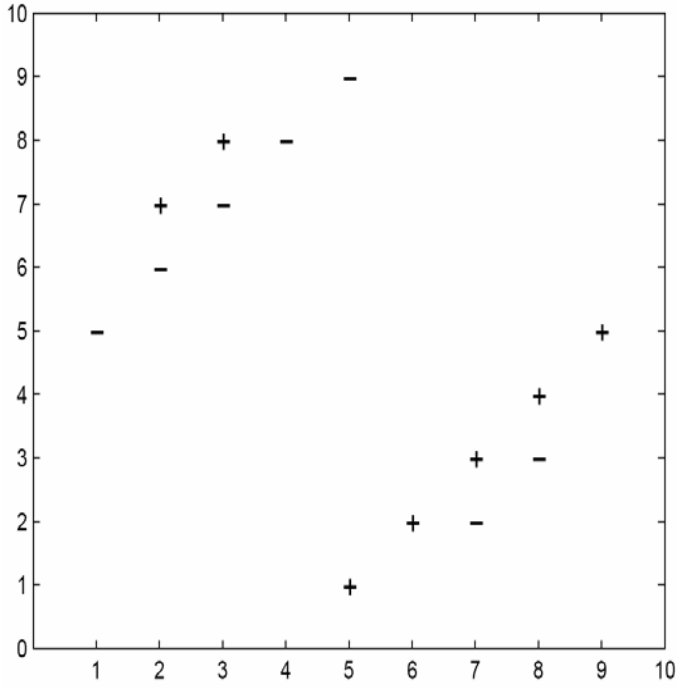
f) Consider the datasets in figures below, A and B. In each of these datasets there are two classes, '+' and 'o'. Each class has the same number of points. Each data point has two real valued features, the X and Y coordinates. For both of these datasets, we want to design a Bayes classifier assuming Gaussian distributed data . The covariance matrices are not equal across classes, but they are diagonal, on the form $\Sigma_j=\sigma^2 I$.

Given these assumptions, sketch the resulting decision boundaries in both cases. If the classifier breaks down, explain.




## Exercise 6. K-nearest neighbor classification

In the following questions you will consider a *k*-nearest neighbor classifier using Euclidean distance metric on a binary classification task. We assign the class of the test point to be the class of the majority of the *k* nearest neighbors. Note that a point can be its own neighbor.



a) In the figure, sketch the *1*-nearest neighbor decision boundary for this dataset.

b) If you try to classify the entire training dataset using a kNN classifier, what value of *k* will minimize the error for this dataset? What is the resulting training error?

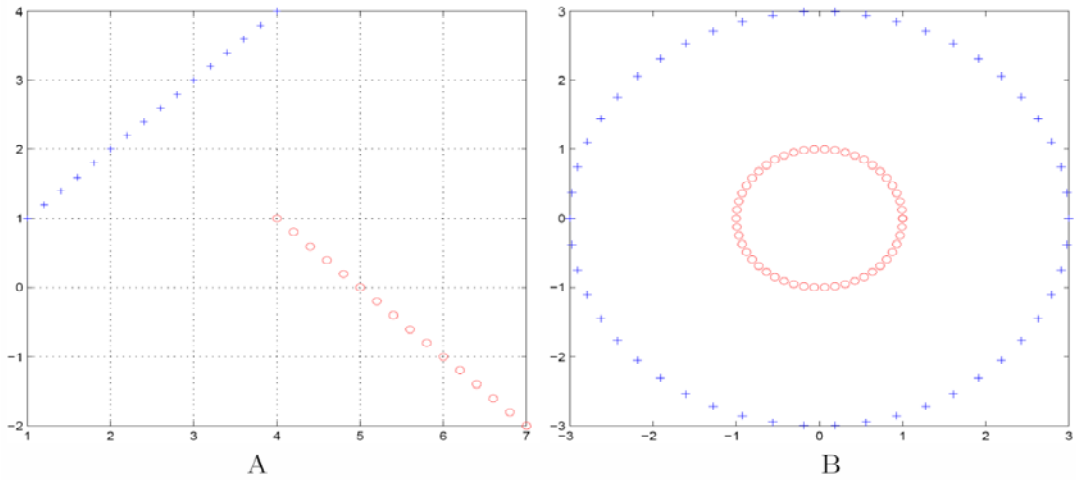
c) What happens if you use a very large $k$ on this dataset? Why might too small values of $k$ also be bad?

d) What value of $k$ minimizes leave-one-out cross-validation error for this dataset? For any reasonable choice of $k$, the resulting minimal error is 4/14.

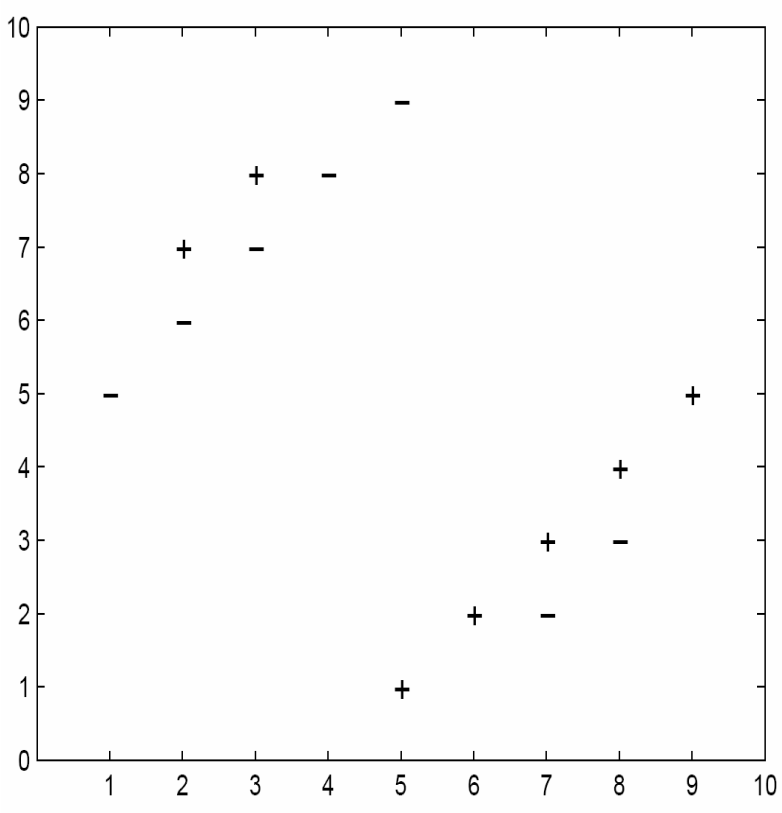
**Candidate number:**

**Enclosure – sketch your answers here**

## Exercise 6. Classification using Bayes rule




## Exercise 7. K-nearest neighbor classification



In the following questions you will consider a $k$-nearest neighbor classifier using Euclidean distance metric on a binary classification task. We assign the class of the test point to be the class of the majority of the $k$ nearest neighbors. Note that a point can be its own neighbor.

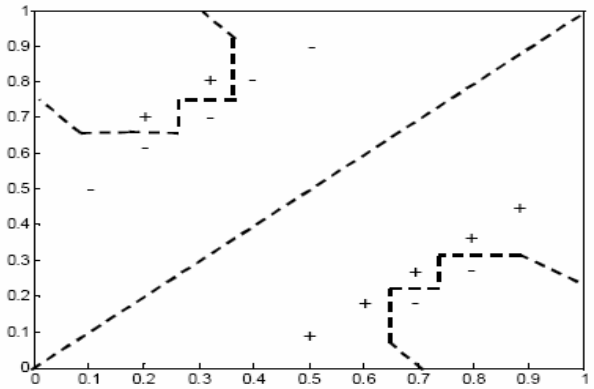



Figure 2: ★ **SOLUTION:** 1-nearest neighbor decision boundary.

1. [3 points] What value of $k$ minimizes the training set error for this dataset? What is the resulting training error?

   ★ **SOLUTION:** Note that a point can be its own neighbor. So, $k = 0$ minimizes the training set error. The error is 0.

2. [3 points] Why might using too large values $k$ be bad in this dataset? Why might too small values of $k$ also be bad?

   ★ **SOLUTION:** Too big $k$ ($k = 13$) misclassifies every datapoint (using leave one out cross validation). Too small $k$ leads to overfitting.

3. [6 points] What value of $k$ minimizes leave-one-out cross-validation error for this dataset? What is the resulting error?