Pattern Recognition 2022-2023.1A (WMAI021-05)

Instructions for Assignment-1

Total points: **100**

Starting date: 9 September 2022
Submission deadline: **11:59PM, 3 October 2022**

General guidelines:

- The tasks are targeted at groups of four students. Please make sure that the load is well divided: every student should contribute. In the case of not equal contributions that grade will be divided accordingly.
- Please take advantage of the tutorial sessions to ask your questions about the tasks.
- Provide a (short but comprehensive) explanation of what you are doing for each (sub) task.
- A reviewer should be able to understand plots independently; be sure to label axes, a legend for colors, use an easily readable font size, etc.
- Refer to all plots, tables, code blocks, etc. in your report.
- For the report: you can use the template (provided in Brightspace).
- Whenever the word *implement* is mentioned, the algorithm in question is to be implemented by the students without just using a direct call to an existing Matlab function.
- Matlab is the mandatory programming language for this first assignment. Matlab can also be accessed online by using a university account to register on the official website .
- For the code: add codes/scripts for individual tasks to folders with the names *Task_TaskNumber*,e.g., *Task_09*. Submit all the task folders in a zipped file named *Code_Group_GroupNumber*,e.g., *Code_Group_20*.

Contact information:

- If you have questions about the tasks, please send an email to:
  patternrecognition.ai(at)rug.nl
- Include your group and assignment number in the subject line. For example:
  "[Grp4-As1] Question about LVQ"

Good Luck!

Task 1: Pairwise correlation coefficients

Maximum obtainable points: **5**

Consider the 24 × 3 array in the file **task_1.mat**. Each row is a three-dimensional (3D) feature vector. The first element of such a vector is the height of a person in centimeters; the second element is the age in years; the third element is the body weight in kilograms.

1. Compute the pairwise correlation coefficients between the features (elements of the vectors). The correlation coefficient of two features is equal to their covariance divided by the square root of the product of their variances.
2. Create 2D scatter plots of the data points with x- and y-axis being:
   (a) plot A: the two features for which the correlation coefficient is largest
   (b) plot B: the two features for which the correlation coefficient is second largest

   What conclusions can you draw from these plots?

Task 2: Hamming distance

Maximum obtainable points: **15**

Before you start this assignment perform these steps, you do not need to include them in your report:

- Consider the two-dimensional binary arrays in files **person01.mat** to **person20.mat**. Each row of such an array person[i].m is a binary feature vector of 30 elements that is extracted from an iris image of a person that we call here person[i] (i = 1...20). Hence, each row is a 30-dimensional binary iris code of that person. There are 20 such iris codes of each person in the corresponding file person[i]; each row of the array is one such binary iris code.
- Take a closer look at the rows of one such array and notice that two rows can differ in only a few positions (bits). Compare now two rows that come from two different files: person[i] and person[j]. Notice that two such iris codes differ in about 15 positions.

Discuss the following points in your report:

1. The Hamming distance (HD) of two binary iris codes is the number of positions (bits) in which the two codes (binary feature vectors) differ. Compute two sets S (Similar) and D (Different) of 1000 HD values each as follows:
   (a) For set S: Choose randomly one of the files person[i].mat, i = 1...20. Choose randomly two rows in that file. Compute the HD of these two rows. Normalize the HD by dividing it by 30. (If you use a Matlab function that computes the normalized HD, you need not divide by 30.) Repeat this process 1000 times to obtain 1000 HD values.
   Hint 1: Use the function **sprintf** to generate the filename for a person. For example **sprintf**('person%02d.mat',3) gives 'person03.mat'.
   Hint 2: Create a string array containing strings 'person01.mat', 'person02.mat' etc, using the function char, to be able to load a random file.
   (b) For set D: Choose randomly two different files person[i].mat and person[j].mat, i = 1...20; j = 1...20; i $\neq$ j . Choose randomly one row from each of these two files. Compute the HD of these two rows. Normalize the HD by dividing it by 30. Repeat this process 1000 times to obtain 1000 HD values.
2. Plot the histograms of S and D in one figure with different colors. Make sure to use bins of the same size for the two histograms and to use an appropriate number of bins. How much do the two histograms overlap?

3. Compute the means and the variances of the sets S and D.

4. Add to the figure of the previously computed histograms plots of two normal distributions (Gaussian functions) with the above computed means and variances. Find an appropriate way to scale the normal distribution curves so that they fit well the histograms. Explain and justify your way of scaling.

5. The distribution associated with the set S is the class-conditional probability density function that measures a given HD value for two iris codes of the same person. The distribution associated with the set D is the class-conditional probability density function that measures a given HD value for two iris codes of two different persons. To compute the decision criterion either use Matlab's **normcdf** function and iterate to find the correct value, or use $\sqrt{2} \cdot$ **erfinv** of the exact confidence interval. The false acceptance rate is the value of the integral of the normal distribution corresponding to the set D for HD $<$ d, where d is the value of the decision criterion. False rejection rate is the value of the integral of the normal distribution corresponding to the set S for HD $>$ d.

   (a) Estimate the value of the decision criterion for which the false acceptance error is 0.0005.

   (b) For that value of the decision criterion, determine the false rejection rate.

   Note that here the terms acceptance (of an impostor) and rejection (of an authentic person) are related to the alternative hypothesis stating that two iris codes which are compared come from the same person, the zero hypothesis being that they come from two different persons. False acceptance and false rejection thus correspond to an error type I and II, respectively, in terms of statistical decision theory and hypothesis testing.

6. Consider the iris code given in the file **testperson.mat**. This file contains an iris code of which some bits are missing. These missing bits have the value 2 instead of 0 or 1. To which of the 20 people whose iris codes are stored in files person01.mat to person20.mat does this iris code most likely belong to?
   Hint: You need to create a mask based on the test person iris code that masks the missing bits and use this mask when comparing the testperson with other iris codes. .

Task 3: Covariance matrix

Maximum obtainable points: **5**

1. Compute the mean and the covariance matrix of the following set of feature vectors: [4 5 6; 6 3 9; 8 7 3; 7 4 8; 4 6 5].
   Hint: We consider a feature vector to be a vector of values of measurements of different features of an object. So here we have five measurements (the five vectors) of three features; the first feature having values 4, 6, 8, 7 and 4, the second feature having values 5, 3, 7, 4 and 6 and the third feature having values 6, 9, 3, 8 and 5.

2. Using the computed mean vector and covariance matrix, model the observed data by a normal distribution and compute the probability density in the points: [5 5 6], [3 5 7] and [4 6.5 1].
   Hint: Matlab already provides a `mvnpdf` (Multivariate normal probability density function) function .

Task 4: 2D Gaussian

Maximum obtainable points: **5**

Generate a two-dimensional Gaussian pdf with a mean [3 4] and covariance matrix [1 0; 0 2].

1. Plot this function on [-10 10] $\times$ [-10 10] using the *mesh* function.

2. Implement and compute the Mahalanobis distance between the points [10 10]', [0 0]', [3 4]', [6 8]' and the mean of this density function.
   Hint: For the implementation use the definition of the Mahalanobis distance from Wikipedia or Mathworld.

Task 5: Naive Bayesian rule

Maximum obtainable points: **5**

	Spam	**Non-spam**
Anti-aging	0.00062	0.000000035
Customers	0.005	0.0001
Fun	0.00015	0.0007
Groningen	0.00001	0.001
Lecture	0.000015	0.0008
Money	0.002	0.0005
Vacation	0.00025	0.00014
Viagra	0.001	0.0000003
Watches	0.0003	0.000004

Table 1: Probabilities of the occurrence of keywords in email

Table 1 presents the probabilities for the occurrence of certain keywords in an email. Assume that the priors of receiving spam and non-spam are 0.9 and 0.1, respectively.

1. Using the naive Bayes rule (either using own Matlab code or manual calculations), classify the following email texts as spam or non-spam:
   (a) “We offer our dear customers a wide selection of classy watches.”
   (b) “Did you have fun on vacation? I sure did!”

Task 6: Decision Tree

Maximum obtainable points: **5**

Consider the dataset below. Given the last column is the class you want to predict, what tree best represents the data? Figure it out manually, thinking what will be the first node/branch and why. Add your calculations to the report.

Size	**Colour**	**Edible**
small	yellow	yes
small	yellow	yes
small	red	no
large	red	yes
small	yellow	yes
large	yellow	no
small	red	no
small	yellow	yes

Task 7: Learning Vector Quantization

Maximum obtainable points: **10**

The files **data_lvq_A.mat** and **data_lvq_B.mat** each contain 100 two-dimensional feature vectors, belonging to class A and B, respectively. Your task is to build a LVQ classifier using this training data.

1. First investigate this data set by making a scatter plot of the two classes, where the classes are distinguishable. How many prototypes per class (at least) would be appropriate to use for this data set?
2. Implement the LVQ algorithm. The gist of the algorithm is that after initialising prototypes, for each point in the training set you choose the prototype that is the closest to this point. If the closest prototype belongs to the same class as the point, you move it closer to the point (respectively of the learning rate η), and if the prototype belongs to a different class, you move it further from the point (again, respectively of the learning rate η). Use a constant learning rate $\eta = 0.01$. Determine the prototype by minimal squared Euclidean distance. Think about how to choose the initial prototypes, discuss the method you used. After each epoch, determine the number of misclassified training examples. The training error E is defined as the number of misclassified training examples divided by the total number of data points. The stopping criterion can be chosen based on E, either by choosing a threshold or by looking for when the value changes very little at each epoch(the value is "constant"). Chose a stopping criterion you find fit but make sure you do not stop too early. Report the number of epochs when the training stops.
3. Consider the following cases:
   (a) 1 prototype for class A and 1 prototype for class B,
   (b) 1 prototype for class A and 2 prototypes for class B,
   (c) 2 prototypes for class A and 1 prototype for class B,
   (d) 2 prototypes for class A and 2 prototypes for class B.

   For each of these cases

   (a) Plot E as a function of the number of epochs. Analyze each of these cases.
   (b) Create a 2D-plot which shows the LVQ prototypes and the train data.

Task 8: Cross-Validation

Maximum obtainable points: **5**

1. For case 3c in the previous task, apply 10-fold cross validation for the estimation of the classification error. This means, divide the training set in 10 equal subsets. Use 9 subsets to train an LVQ classifier and then use the remaining 1 subset to test it and determine the classification error. Use each of the subsets as a test set once, with the other 9 subsets as training sets.
2. Compute the test error, i.e. the mean of the 10 values of the classification error computed in this way.
3. Make a bar plot of the final training error for each fold (10 bars). Add the value of the test error you found at question 1 as a horizontal line over the bars in the bar plot. Make sure the plot contains the numerical values for the errors (in percentages) for each bar.

Task 9: ROC

Maximum obtainable points: **5**

Consider two normal distributions with different means $\mu_1 = 5$ and $\mu_2 = 7$ ($\mu_1 < \mu_2$) and equal variances $\sigma^2 = 4$ (see Fig. 2.19 in page 49 of Pattern Classification book by R. Duda, or lecture slides from week 3). Let x^* be the value of a decision criterion used to classify an object having a feature with value x in class ω_1 for $x < x^*$ and in class ω_2 for $x \geq x^*$.
The integral of the first distribution (with mean μ_1) for values of $x \geq x^*$ specifies the probability of wrongly classifying an object from class ω_1 into class ω_2 to be referred to as a false alarm. The integral of the second distribution for values of $x \geq x^*$ specifies the probability of correctly classifying an object from class ω_2 into class ω_2, to be referred to as a hit.

1. Choose a value of x^* and compute the probabilities of hit (h) and false alarm (fa). Plot the point (fa, h) in a graph with horizontal axis fa and vertical axis h. Choose a few other values of x^* in the interval $[\mu_1 - 3\sigma; \mu_2 + 3\sigma]$ and plot the corresponding (fa,h) points too. Connect the points with a curve, which is called the ROC curve. This curve corresponds to discriminability:

$$d' = \frac{|\mu_2 - \mu_1|}{\sigma} = \frac{|7-5|}{2} = 1. \tag{1}$$

   Repeat the computation of a ROC curve for the cases $\mu_2 = 9$ and $\mu_2 = 11$ and plot all three ROC curves in the same diagram. What is the value of the discriminability d for each of these cases?

2. Consider the binary vectors given in file **task_9.mat**. They specify the outcomes of a psychometrical experiment in which a test person is presented with a visual stimulus that contains or does not contain a given signal and the test person has to specify whether he/she detected the signal. The vectors code for the following outcomes:

1 1	TP	the signal was presented and detected (hit)
1 0	FN	the signal was presented but the person failed to detect it
0 1	FP	the signal was not presented but the test person indicated to have detected it (false alarm)
0 0	TN	the signal was not presented and the test person indicated that there was no signal

Compute the values of the hit rate h and the false alarm rate fa and plot the point (fa,h) in the plot computed in Task 9.1 above. This point lies on a ROC curve with a given discriminability value d^0. Determine this value by trial and error, i.e. taking different values d^0 and drawing the corresponding ROC curves until you find a value of d^0 for which the corresponding ROC curve passes through the experimentally determined point.

Hint 1: Notice that in this case, we do not know the values μ_1, μ_2, σ and x^* of the detecting system. And yet we can compute its discriminability d^0 using the empirically obtained hit and error rates (and under the assumptions that the internal distributions are normal and have equal variances, the difference in their means is caused by the presence of a signal in the visual stimulus).

Hint 2: ROC curves have the same shape regardless of the exact underlying data as long as you keep in mind that the false alarm distribution should have a bigger mean than the hit rate distribution and that they both should have the same sigma. So just create artificial distributions, plot them and see for which values your ROC curves go the closest to the point (fa,h).

Hint 3: Note that the first feature encodes whether or not the signal was shown, and the second feature denotes whether or not the test person claims to detect something. So the hit rate is defined as the probability that the test person claims to detect something, given that a signal was presented. And the false alarm rate is the probability that the test person claims to detect something, given that no signal was presented.

Task 10: Transforms

Maximum obtainable points: **15**

Part 1:

In this exercise you will apply the matlab function `hough`.

1. Read and show the image **cameraman.tif**. This image is supplied by matlab.
2. Compute the edge map of the image using the Canny algorithm with the matlab default parameters. Show this edge map.
3. Compute the Hough transform accumulator of the edge map.
4. Plot the accumulator array.
5. Threshold the image to keep only the strongest responses of the accumulator array. Define your own definition of what the strongest responses are.
6. Find the local maxima in the thresholded accumulator.
7. Plot the accumulator array and mark the five strongest local maxima points on it.

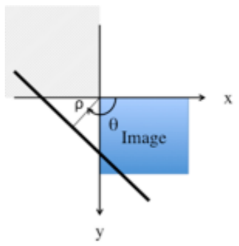

Figure 1: Image in spatial coordinates (image space), with the associated ρ and θ.

8. Show the original image, cameraman.tif, and overlaid the strongest line. Create a new function `myhoughline`, that takes as inputs an image, ρ and θ and outputs a figure in which a line is drawn at perpendicular distance ρ from the upper left corner of the figure, with angle θ from the top horizontal edge of the image to the perpendicular.

Hint: Create a special case for vertical lines.

Part 2:

Suppose you will write your own matlab function `myhough` that uses the Hough transform based on parameterizations of the type

$$\rho = xcos\theta + ysin\theta, \tag{2}$$

where ρ and θ are respectively the orientation and distance to the center of a line.
Decide on a discrete set of values of θ and ρ to use. The usual convention is to take values of θ in the range $-\frac{\pi}{2} \le \theta \le \frac{\pi}{2}$ and let ρ have both positive and negative values. However, for the sake of simplicity, we are going to use the values of θ and ρ as indicated in the accumulator array by restricting ρ to non-negative values. Figure 1 represents an image in spatial coordinates (image space). Restricting ρ to positive values, which values of θ should be considered to represent lines in an image?

Part 3:

In this exercise we will use the built-in Matlab functions to understand the Hough transform for lines.

1. Create a 50×50 black image with a single white pixel. Compute the Hough transform of the image. Show both the image and the Hough space of the image.
2. Create a $50{\times}50$ black image with three non-aligned white pixels. Compute the Hough transform of the image. Show both the image and the Hough space of the image.
3. Create a 50×50 black image with three aligned white pixels. Compute the Hough transform of the image. Show both the image and the Hough space of the image.
4. Compare and explain the results obtained in exercises 1, 2 and 3.
5. Use houghpeaks to find the maxima in the Hough space from the image with the three aligned points. Show the maximum peak in the Hough space.
6. Extract the line from the houghspace of the image with three aligned points using houghlines and show it in the original image, marking the beginning and end of the line.

Part 4:

In this exercise we will use the built-in Matlab functions to understand the Hough transform for circles. Our aim is to find the circular screws of an industrial image of a milling machine.

1. Read the image **HeadTool10002.bmp** and convert it to double precision.
2. Apply the contrast-limited adaptive histogram equalization method to enhance the contrast of the image.
3. Find at least 6 circles, if two circles belong to the same screw, keep only one of them. **Hint**: Try radii between 20 and 40 pixels, and different values of sensitivity with `imfindcircles`.
4. Show the enhanced image from exercise 2 with the circles found in exercise 3.
5. Show the enhanced image from exercise 2 with the two strongest circles found in exercise 3.
6. Show the accumulator array peaks.

Task 11: K-Nearest Neighbor

Maximum obtainable points: **10**

We want to study *K-nearest neighbor* for classification. Given is the data file **task_11.mat** containing 200 2D data points in the space $[0, 1] \times [0, 1]$. The first 100 points belong to class ω_1 and the second 100 points belong to class ω_2. The program in the file **knn_wrapper.m** uses such a classifier. For each point in the space, it determines the class by K-nearest-neighbor classification. The resulting grayscale image shows to which class each point in the space belongs (in the case of two classes, black is ω_1, and white is ω_2).

1. Implement the KNN function with the Euclidean distance function and give the code in your report. For a given K it should return the class to which point (X, Y) belongs based on the data be one of the class_labels. Write your function in such a way that it works with more than two classes (see point 4) and with more than two features as well (i.e. given [X Y Z] instead of [X Y] as the first parameter (the dimensions of data will then, of course, be different as well).
2. Show the results for classification if K = 1, 3, 5, 7.
3. Determine the optimal choice of the parameter K in the range 1, 3, ..., 25 using leave-one-out cross-validation:

- For each point in the data set, make a copy of the dataset without that point.
- Classify the point using the reduced dataset.

Report the error rates for the different values of K (in a plot or a table), which one is the best? And can you explain this result?
Hint: Use array(i)=[] to remove an element from an array.

4. Repeat 2 and 3 but now assume that there are 4 classes containing the points with indices (1:50, 51:100, 101:150, 151:200).
Now in the case where K = 3, for instance, it may happen that all three nearest neighbours are of a different class (similar scenarios exist for higher K). In case of such a tie, make an arbitrary choice. You can for example simply choose the class with the lowest class number.

Task 12: K-means clustering

Maximum obtainable points: **15**

1. The file **kmeans1.mat** contains 2D feature vectors. Apply the k-means clustering algorithm with Euclidean distance to this data for the k values given below. As a stop criterion use the condition that none of the data points are re-assigned to a different cluster. After completion of the algorithm for a given value of k, make a scatter plot of the data points, taking care that the different clusters can easily be distinguished, e.g. by rendering points that belong to different clusters in different colors or shapes if you print in black and white. Add the final cluster means to the same plot, making sure they are distinguishable from the data points. Make another plot that shows all intermediate positions of the cluster means computed at the different iterations of the algorithm, indicating which are the starting and ending position of each cluster mean.

   Make the steps described above for k = 2, 4, 8 means and include the resulting images in your report.

   Hint: Write your algorithm as a function that takes the 2D array of feature vectors and the number k of means as arguments so that you need not change your code to use it for a different number of means or a different dataset. Initialize the k-means as k distinct points randomly chosen from the dataset. You can make use of the provided plot arrow function to illustrate the movement of the means.

2. The k-means algorithm can get stuck in local minima, and so the quality of its results depends heavily on the initialization of the cluster-means. The k-means++ algorithm includes a specific procedure for initializing the prototypes of the k-means algorithm such that its solution is close to the optimal k-means solution.
The k-means++ algorithm is as follows:

   1) Choose one point at random from the set of all data points and let this point be the first prototype.
   2) For each data point x, compute D(x), that is the distance between x and the nearest prototype. (In the beginning, there is only one prototype but later other prototypes are being added, see below.)
   3) Choose one new data point at random as a new prototype, using a probability distribution where a point x is chosen with probability proportional to $D(x)^2$. Thus, a point x is more likely to be a prototype if it's further from already existing prototypes.
   4) Repeat Steps 2 and 3 until k prototypes have been chosen.
   5) Now that the initial prototypes have been chosen, proceed with the standard k-means clustering algorithm.

a) Modify your k-means function to require a third parameter signifying whether k-means++ initialization should be used. Implement k-means++ initialization in your k-means function.

b) The file **checkerboard.mat** contains 3700 2D feature vectors. For this data and using k = 100, compute the mean of the quantization error and its standard deviation averaged over 20 runs of the k-means algorithm, with and without using k-means++.

c) Does the use of k-means++ initialization for this dataset lead to a statistically significant improvement?
The context of this question is as follows: for each run of the k-means and the k-means++ algorithm you may get different values of the quantization error. In general, we expect that the k-means++ algorithm will yield a smaller quantization error but it may happen that for some run the quantization error obtained with the simple k-means algorithm is smaller than the quantization error achieved with the k-means++ algorithm for the same or for some other run. If the average value of the quantization error obtained with the k-means++ algorithm over the considered 20 runs is smaller than the corresponding average value obtained with the k-means algorithm, does this mean that this will still be the case if we perform another 20 runs with the two algorithms? The certainty with which we can say that the k-means++ algorithm outperforms the k-means algorithm depends on the difference of the two average values in relation to the corresponding standard deviations (spreads). It can be computed and expressed as a p-value using an unpaired one tailed two-sample t-test (e.g. Welch's t-test). Perform such a test and determine the concerned p-value.