EXAMINATION PAPER
TAKE-HOME EXAM

Take-Home exam in	:	FYS-3012/8012 Pattern Recognition
Hand out	:	9 AM Monday October 16$^{\text{th}}$ 2023
Hand in	:	2 PM Monday October 30$^{\text{st}}$ 2023

Contact person	:	Anthony Doulgeris
Telephone	:	776 45177
Email:	:	anthony.p.doulgeris@uit.no

The exam contains 6 pages including the front page.

The exam uses an auxiliary file: `ExamData3D.mat`

Instructions

- The report must be submitted as one single file in pdf format. No exceptions from this will be allowed. You should use your candidate number to identify yourself, not your name.

- Source code in Python, Matlab or another programming language of choice shall be included in the report, but mostly at the end in an Appendix. Any code that is explicitly discussed in the text can be inserted in the text, but most should be placed in the appendix. Runnable code may also be submitted as a zipped attachment/auxiliary file. The source code must be commented according to good programming practice.

- All figures that you produce while solving the exercises shall be included in the report. These shall be annotated by labelled axes, captions and legends explaining the graphs when it is appropriate.

- Explain your approach to solving the problems, the assumptions you make, and how you reason. Lack of explanations will result in a lower score.

- If you can't actually solve a problem, due to time or programming problems, you may want to write down some notes about how you *would have* done it. Your intentions may gain you some partial score for the question.

- Some exercises may require you to make your own implementational choices. That is, the implementation of the algorithms may not be completely specified. In such cases, state your choice and argue for it. It does not need to be a lengthy argument.

- The take-home exam must be answered individually. Any signs of collaboration, transcription or plagiarism will be investigated and prosecuted.

- The take-home exam counts 50% towards the total grade in FYS-3012. According to the regulations, if you fail this Take-Home Exam, then you automatically fail the course, in which case the student will not be be granted access to the final exam in December.

- All lettered subquestions will count equally towards the grade of the report, e.g., (a), (b), (c), etc., even if some have bulleted sub-tasks.

BACKGROUND

• *This exam works with a synthetic multi-class data set in 3-dimensions and will explore several different aspects from the course. You will uses Bayesian methods, linear classifiers, non-linear Neural Networks and feature selection.*

• *You are given a data file* `ExamData3D.mat` *containing both a training set* (`X_train` *and* `Y_train`) *and an independent testing set* (`X_test`, `Y_test`). *The data file is saved in a MatLab* `.mat` *format that can be read in MatLab with the* `load()` *function and in Python with the* `scipy.io.loadmat()` *function.*

• *Several different classification methods will be tested and compared at the end. You should evaluate the number of training data errors, the number of testing data errors and the test data classification accuracy, as a percentage, for each to aid in your comparison.*

• *You should use your own code wherever possible, and most routines were covered during the exercise classes. If you did not complete these codes, and cannot do so now, then you may use any toolbox or existing code instead but you must reference where you got it.*

1 FEATURE SELECTION

Here you will explore the features, individually and in combinations, to work out their value for the purpose of classification. We shall use a modified J3 score, that has been normalised by subtracting the expected minimum. Start with the textbook definitions for the scatter matrices, S_w , S_b, $S_m = S_b + S_w$ and $J_3 = \text{trace}(S_w^{-1} S_m)$.

(1a) Calculate the expected minimum value for J_3 when all classes are identical, consider the general multi-dimensional case. Give your reasoning or show any calculations with the defined expressions.
Use this minimum expression to modify your

$$J_3 = \text{trace}(S_w^{-1} S_m) - J_{\min}$$

and use this new J_3 hereafter.
Can you suggest which situations this modification will be important for comparing J3?

(1b) Your data is 3-dimensional and gives seven combinations, $\{1, 2, 3, [1, 2], [2, 3], [1, 3], [1, 2, 3]\}$. Plot each of the 1-D histograms and 2-D scatter plots, with a different colour for each class, but don't try to plot the 3-D figure.
Calculate and put the modified J3 score in the title of each plot, and also calculate and note the full 3-D J3 score for comparison. The scores should be small when there is little class separability and high when there is more separability.

(1c) Does this correspond with what you see?
Which 2-D and 1-D *feature selections* does it indicate are best for classification?
Does the full 3-D score add any extra value? Why or why not?

• *If you find good reasons to reduce to 2 dimensions in this feature selection exercise (but not to one dimension, that is too simple), then you may do so in the subsequent questions. If not, or if unsure, then continue to use the full 3-D data-set.*

2 BAYESIAN METHODS

Bayesian decision theory is all about using probability density functions to choose the most likely class. We have training data, so we can estimate densities. Careful inspection of the previous plots and histograms will show that the classes are not simple Gaussians, some are skewed and kinked. (For your information, some are generated as a mixture of two Gaussians.)

(2a) For a reference, first classify the image using a maximum likelihood multi-variate Gaussian classifier. That is, assume that the classes are Gaussians, estimate the three sets Gaussian parameters (for each class) from the training data, and then classify the test data. Calculate the number of errors for the training set and testing set separately, and the final test set classification accuracy.

(2b) Briefly explain the main principle behind a Parzen window density estimation, also known as kernel density estimation (KDE). Consider only a Gaussian Kernel.
How can a symmetric Gaussian kernel end up estimating non-Gaussian, non-symmetric distributions?
What choices are involved, and what effect do such parameters have?

(2c) Code a Parzen window based classifier, using the training set to classify the testing set. Calculate the training and testing errors and classification accuracy for this approach.

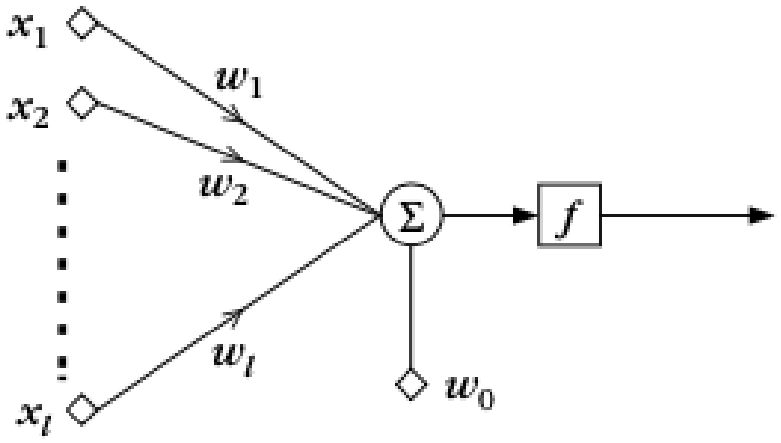
(2d) Find a way to plot the decision regions in dimensions 1 and 2, for a tight region around the actual data samples. For example, using different background shaded colours or by drawing the decision boundaries, and then plot the test data over the top (scatter plot). MatPlotLib has a function called `Contourf()` that may help. Show this figure for both the Gaussian case and the Parzen window case.

(2e) Discuss the appearance of the figures, the number of errors and any reasons for this behaviour.

3 LINEAR PERCEPTRON

This is a 3 class problem in 3 dimensions with 600 training samples. The training data are labelled as classes 1, 2 and 3.
According to Bayes decision theory, you will need three $g_j(\mathbf{x}) = \mathbf{w}_j^T\mathbf{x}+\mathrm{w}_{0j}$ discriminant functions, one for each class, and choose the label of the one with the maximum output value.

(3a) Describe all the parts of a single Perceptron node (pictured) of a multi-layer Perceptron Neural Network.
Indicate the connection to the discriminant function $g_j(\mathbf{x})$ described above?



(3b) Explain how you may code this 3-class linear problem as a Neural Network that only has the three output nodes, with a sigmoid activation function giving a $[0, 1]$ range, and the squared error cost function.
How can we interpret the value of each output node and then combine to choose the final label?
What do you have to do to the training labels to calculate the three node, $[0, 1]$ range, output error for back-propagation?

• *This is different from the two-class output case, where there was only one node, so you may have to modify your previous codes to achieve this multi-class approach if you have not done so already.*

(3c) Code and train this single-layer, 3-class, Perceptron network with the training data. You may choose a fixed number of epochs and training rate determined through trial and error. Plot its decision regions or boundaries along with the data.
Note the accuracy as before with testing and training errors and classification accuracy.

(3d) Discuss the result and compare to the Bayesian methods of section 2.

4 NON-LINEAR NEURAL NETWORKS

Here we will apply the textbook standard multi-layer Perceptron (MLP) Neural Network to the task. Use the sigmoid activation function and the three output layer nodes and the squared error cost function as in the multi-class linear Perceptron problem above.

(4a) Explain the purpose of the learning rate and momentum, and their effect on the convergence behaviour. Show the update expressions with momentum.

Experiment with different design architecture, i.e., number of nodes and layers, and tuning of the learning rates for this problem, to work out what works and what doesn't.
For each example shown in the following questions:

- plot the region or boundary map with the test data over the top
- plot the convergence, learning curves for both the training and testing data
- calculate the training errors, the testing errors and test data classification accuracy.

(4b) Show some examples that demonstrate the effect of network complexity, from very few hidden nodes (e.g. 3-5) to many (e.g. 10-20).
Can you make, and explain, any generalisations relating to the visual effect of complexity?

(4c) For a fixed complexity (your choice), show some examples that demonstrate the effect of the learning rates, both good and poor choices, discussed in (4a).
Try to explain why they look they way they do.

(4d) Finally, describe and show your best design choice and why you think this is good for the task.
Compare and discuss all the different methods that you have now used to classify the data. Refer to the training and testing errors and classification accuracy for each of the Gaussian, Parzen, Linear Perceptron, and multi-layer Perceptron methods, as well as all their region pictures.
Which approach was best? Which was fastest? Which was easiest to code and use? Which do you like the most?