CA Assignment 1
Data Classification
Implementing Perceptron algorithm

Assessment Information

Assignment Number	1 (of 2)
Weighting	15%
Assignment Circulated	10 Feb 2023
Deadline	3 March 2023 at 17:00
Submission Mode	Electronic via Canvas
Purpose of assessment	The purpose of this assignment is to demonstrate: (1) the understanding of the Perceptron algorithm; (2) the ability to implement the Perceptron algorithm for binary classification; (3) the ability to evaluate a classification algorithm; (4) the ability to turn a binary classification algorithm to a multi-class classification algorithm using the 1-vs-rest approach; (4) the ability to incorporate regularisation into a classification algorithm.
Learning outcome assessed	(1) A critical awareness of current problems and research issues in data mining. (3) The ability to consistently apply knowledge concerning current data mining research issues in an original manner and produce work which is at the forefront of current developments in the sub-discipline of data mining.

Objectives

This assignment requires you to implement the Perceptron algorithm using the Python programming language.

Assignment description

Download the *CA1data.zip* file. Inside, you will find two files: *train.data* and *test.data*, corresponding respectively to the train and test data to be used in this assignment. Each line in the file represents a different train/test instance. The first four values (separated by commas) are feature values for four features. The last element is the class label (class-1, class-2 or class-3).

Questions/Tasks

1. **(15 marks)** Explain the Perceptron algorithm (both the training and the test procedures) for the binary classification case. Provide the pseudo code of the algorithm. It should be the most basic version of the Perceptron algorithm, i.e. the one that was discussed in the lectures.
2. **(30 marks)** Implement a binary perceptron. The implementation should be consistent with the pseudo code in the answer to Question 1.
3. **(15 marks)** Use the binary perceptron to train classifiers to discriminate between
   - class 1 and class 2,
   - class 2 and class 3, and
   - class 1 and class 3.

   Report the train and test classification accuracies for each of the three classifiers after training for 20 iterations. Which pair of classes is most difficult to separate?
4. **(30 marks)** Explain in your own words what the 1-vs-rest approach consist of. Extend the binary perceptron that you implemented in part 3 above to perform multi-class classification using the 1-vs-rest approach. Report the train and test classification accuracies for the multi-class classifier after training for 20 iterations.
5. **(10 marks)** Add an ℓ_2 regularisation term to your multi-class classifier implemented in part 4. Set the regularisation coefficient to 0.01, 0.1, 1.0, 10.0, 100.0 and compare the train and test classification accuracies. What can you conclude from the results?

Submission Instructions

Submit via Canvas the following **two** files (**please do NOT zip files into an archive**)

1. the source code for all your programs (**do not provide ipython/jupyter/colab notebooks, instead submit standalone code in a single .py file**), and
2. a PDF file (report) of **no more than 3 pages** providing the answers to the questions.

It is extremely important that you provide the two files described above and not just the source code!

Important notes

(read carefully and double check compliance before submission)

1. No credit will be given for implementing any other type of classification algorithm or using an existing library for classification instead of implementing it by yourself. However, you are allowed to use
   - numpy library for accessing data structures such as numpy.array;
   - random module; and
   - pandas.read_csv, csv.reader, or similar modules **only** for reading data from the files.

   However, it is not a requirement of the assignment to use any of those modules.
2. Your program
   - should run and produce **all results for Questions 3, 4, and 5** in one click without requiring any changes to the code;
   - should output only the required data in a clearly structured way; it should NOT output any intermediate steps;
   - should assume that the input files are named 'test.data' and 'train.data', and are located in the same folder as the program; in particular, it should NOT use absolute paths.
3. Programs that do not run will result in a mark of zero!
4. Your code should be as clear as possible and should contain only the functionality needed to answer the questions. Provide as much comments as needed to make sure that the logic of the code is clear enough to a marker. Marks may be deducted if the code is obscure, implements unnecessary functionality, or is overly complicated.
5. You are allowed to shuffle the data. If you use module random to shuffle the data, use a fixed seed value so that your program always produces the same output. This output should be exactly the one that you provide in the PDF report.
6. Your answers in the PDF report should be succinct, but complete and clear. The clarity and presentation of the report will be assessed.
7. Your submission should be **your own** work. Do not copy or share! Make sure that you clearly understand the severity of penalties for academic misconduct (https://www.liverpool.ac.uk/media/livacuk/tqsd/code-of-practice-on-assessment/appendix_L_cop_assess.pdf).