Machine Learning, 2018-19

Coursework Part I

Description

The first part of your coursework will be based on your implementation of k-Nearest Neighbours and nested cross-validation. The coursework is heavily based on your lab work, so firstly make sure you complete the previous labs (up to and including **Lab 4**).

For the coursework, please make sure to implement **your own code** and not use libraries. You will need to present your own code that performs nested cross-validation and the k-nearest neighbour algorithm, build confusion matrices, and estimate distances between data samples.

Plagiarism: please make sure that the material you submit has been created by you. Any sources you use for code should be properly referenced. Your code will be checked for plagiarism using appropriate software.

Deliverables

The deliverables for your coursework should include:

1. A jupyter notebook, including the code you are submitting for the assignment.
2. A report (preferably PDF) that contains a write-up on your implementation, and answers for a set of free-form questions (more details below).

The tasks for your coursework are based on the IRIS dataset, as used in the labs. Download the coursework notebook that will guide you to complete the tasks for the coursework.

Note: Your code should run without issues on lab machines!

Report

Your report should include the following points discussed below. I'm not giving a specific page length limit, but it shouldn't be longer than a couple of pages.

1. Write-up of your implementation. Include any details on your implementation of cross-validation and k-NN, details on the distance functions you used, and notes on any other code you developed for the coursework. Highlight important points in your coursework here.

2. Results. Your report should contain:

- Tables showing overall test accuracy and standard deviation over 5-folds for (i) clean data[1], and (ii) noisy data, while also indicating parameter choice (number of neighbours, distance function).
- A total **confusion matrix** over the 5-fold cross-validation for each of the runs (i.e. one matrix for clean data, and another for noisy data). Each matrix can be the sum of the matrices for each fold. There should be rows and columns for each class.

[1] See example in *Table 1* - the standard deviation (symbolized using ± in the table) is obtained via the numpy function *std*. The total error is the average error over the 5 folds.

Clean data	accuracy	k	distance
Fold1	80	…	…
Fold2	78	…	…
Fold3	75	…	…
Fold4	80	…	…
Fold5	97	…	…
total	82 ± 0.6	(empty)	(empty)

Table 1. Example results table for clean data.

- **Results Analysis**. Any conclusions you have by observing the behaviour of the algorithm on clean and noisy data. You should explicitly try to answer the following questions, justifying your answers based on the results.
    (a) Do the best parameters change per fold? Can you say that one parameter choice is better regardless of the data used?
    (b) Does the best parameter choice change depending on whether we use clean or noisy data? (Answer for both distance function and number of neighbours.)

3. Questions. Your report should also answer the following questions:

(a) **Exploratory Data Analysis.** What do you observe by plotting the figure for data without noise? What do you observe when you add Gaussian noise and plot again?

(b) **Tie breaking.** Assume that you have selected the number of neighbours to be an even number, e.g., 2. For one of the neighbours, the suggested class is 1, and for the other neighbour the suggested class is 2. How would you break the tie? Write example pseudocode that does this.

(c) **Improving performance on noisy data.** The performance of k-NN on the noisy data should be worse than on the clean data. Suggest at least one way of improving the performance on the noisy data. Try to elaborate on your idea as much as possible, including pseudocode where possible.