CS3920/CS5920 Assessed Coursework Assignment 1

October 10, 2022

This assignment must be submitted by 31 October 2022, 16:00. Feedback will be provided within fifteen working days of the submission deadline.

Learning outcomes assessed

Be able to implement machine-learning algorithms, using the Nearest Neighbours algorithm as an example. Have an understanding of ways to apply the ideas and algorithms of machine learning in science and technology.

Instructions

The coursework assignment must be completed strictly individually. You should not use in your submission any downloaded code or existing implementations of the learning algorithms. The submission is entirely electronic. You should submit one file (a Jupyter notebook) via the course's Moodle page.

Give your Jupyter notebook a reasonable name, such as `NN.ipynb`. It should contain your code for the Nearest Neighbour method, your code for another method (conformal predictor or 3 Nearest Neighbours), and your report containing the numerical results and, optionally, a discussion. Remember that you should submit only one file, a Jupyter notebook; the notebook, however, may (and should) contain different kinds of cells: code, text ("markdown"), and headings.

The file that you submit cannot be overwritten by anyone else, and it cannot be read by any other student. You can, however, overwrite your submission as often as you like, by resubmitting, though only the last version submitted will be kept. Submissions after the deadline will be accepted but they will be automatically recorded as late and subject to the College Regulations on late submissions. Please note that all your submissions will be graded anonymously; your name should not appear anywhere in your submission.

The deadline for submission is **Monday, 31 October 2022, 16:00**. No extension is possible.

> **Note:** All the work you submit should be solely your own work. Coursework submissions are routinely checked for this.

Coursework

This assignment requires an implementation of Nearest Neighbours methods for classification. The methods should be implemented in Python and submitted as a Jupyter notebook. Please leave all output produced by the system (i.e., do not remove the contents of cells like "Out [1]:").

Datasets

This assignment uses two datasets. One dataset is `iris` and another, called `ionosphere`, can be downloaded from the course's Moodle page.

- You are already familiar with the `iris` dataset. Each sample has 4 features describing sepal length, sepal width, petal length, and petal width of an iris plant. There are three possible labels, $\mathbf{Y} = \{0, 1, 2\}$.
- The file `ionosphere.txt` contains data collected by a radar system in Goose Bay, Labrador. This system consists of a co-phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. Each line of this file represents one labelled sample with comma-separated features. The last number in the line describes the classification. "Good" $(+1)$ radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" (-1) returns are those that do not.

When dividing the datasets into training and test sets, you may rely on the default convention in `scikit-learn`.

The datasets are based on:

- A. Frank and A. Asuncion (2010). UCI Machine Learning Repository http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science.

For each dataset do the following:

1. Load it into Python, e.g., using `load_iris` for `iris` and `genfromtxt` for `ionosphere`.
2. Split the dataset into the training and test sets. You may use the function `train_test_split` in `scikit-learn`, in which case use your birthday in the format DDMM as `random_state` (omit leading zeros if any).
3. Implement the Nearest Neighbour method (i.e., the 1 Nearest Neighbours method). Run it on both datasets: train it on the training set and compute the number of errors it makes on the test set and the test error rate (the ratio of the number of errors to the size of the test set).
4. Implement the conformal predictor based on the Nearest Neighbour conformity measure

$$\frac{\text{the distance to the nearest sample of a different class}}{\text{the distance to the nearest sample of the same class}}.$$

   Alternatively, you may implement the 3 Nearest Neighbours algorithm (which, however, will not allow you to get 100% for your work: see below).
5. All results should be given in your Jupyter notebook (see below for a description of what to include).

Details of implementation

The Nearest Neighbour and K Nearest Neighbours algorithms are described in Chapter 2. For Nearest Neighbour, the predicted label for a test sample x^* is the same as the label of the nearest training sample (in Euclidean metric $\|x - x^*\|$). For the 3 Nearest Neighbours algorithm, the predicted label is obtained by majority vote among the three nearest neighbours (it is up to you how to treat potential ties). You are not allowed to use any existing implementations. Please also try to avoid using powerful functions such as `np.sort`, and especially applying them to large arrays (such as sorting the distances to all training samples, which would make your task easier but lead to inefficient code).

Calculate the predicted labels for all test samples and compare them with the true labels for the test samples. Calculate the percentage of correct predictions for both datasets. Give the results in your Jupyter notebook.

If you opt for implementing the conformal predictor, calculate the p-values for all possible labels for all test samples and then calculate the average false p-value. Give the results for both datasets in your Jupyter notebook. Please be careful with division by 0, and especially with $0/0$. The standard definitions are $a/0 = \infty$ for $a > 0$ and $0/0 = 0$. There are several ways to get ∞ in Python; e.g., you can use `math.inf` (after importing the `math` module, `import math`). Or you can import ∞ from `NumPy`: `from numpy import inf`.

Numbers to include

Therefore, you should give the following 4 numbers in your Jupyter notebook (at some prominent place):

- If you choose to implement the K Nearest Neighbours algorithm for $K = 1$ and $K = 3$, you should give the test error rates for:
  - The `iris` dataset and $K = 1$;
  - The `iris` dataset and $K = 3$;
  - `ionosphere.txt` and $K = 1$;
  - `ionosphere.txt` and $K = 3$.

  The maximal mark you can get if you use this option is 85% (plus the extra marks, if any, as described below).
- If you choose to implement the Nearest Neighbour algorithm and conformal predictor, you should give:
  - the test error rate for Nearest Neighbour applied to the `iris` dataset;
  - the test error rate for Nearest Neighbour applied to `ionosphere.txt`;
  - the average false p-value for the Nearest Neighbour conformal predictor applied to the `iris` dataset;
  - the average false p-value for the Nearest Neighbour conformal predictor applied to `ionosphere.txt`.

The maximal mark you can get if you use this option is 100%.

Marking criteria

To be awarded full marks you need both to submit correct code and to obtain correct results on the given datasets. Even if your results are not correct, marks will be awarded for correct or partially correct code. An ideal implementation of the Nearest Neighbour algorithm will give you 65%. If, in addition, you implement the 3 Nearest Neighbours algorithm, this will give you another 20%. If, in addition to Nearest Neighbour, you implement the Nearest Neighbour conformal predictor, this will give you another 35% (in this case there is no need to implement the 3 Nearest Neighbours algorithm).

Every year we get a few submissions where students simply experiment with `scikit-learn` functions, without providing their own implementation of Nearest Neighbours. In practice, they get a few marks (at most 40%) for this.

Extra marks

There are several ways to get extra marks (at most 10%) that will be added to your overall mark (the sum will be truncated to 100% if necessary). Extra marks will be given for:

- Implementing the Nearest Neighbour conformal predictor in a computationally efficient way, avoiding (at least some) superfluous computations.
- Checking the validity of your conformal predictor, along the lines of Lab Worksheet 4, Section 4 (I think this is feasible only for a computationally efficient conformal predictor).
- Experiments with different conformity measures (such as those given in Chapter 3, slide 15).
- Justifying your convention for $0/0$.
- Implementing the K Nearest Neighbours algorithm for a general K.
- Any other improvements of interesting observations about the methods or the datasets (discuss these in your Jupyter notebook).