



## Assignment 1 (12 pts)

Due February 8, 2021, 23:59 EST

*Read the `PythonSetup.pdf` document (posted on Quercus) before beginning this assignment.*

- Q1) 4pts** Implement the k-NN algorithm for regression with two different distance metrics ( $\ell_2$  and  $\ell_1$ ). Use 5-fold cross-validation<sup>1</sup> to estimate  $k$ , and the preferred distance metric using a root-mean-square error (RMSE) loss. Briefly describe your search procedure to estimate  $k$  and the distance metric. Compute nearest neighbours using a brute-force approach.

Apply your algorithm to all regression datasets (use `n_train=1000`, `d=2` for `rosenbrock`). For each dataset, report the estimated value of  $k$  and the preferred distance metric, and report the cross-validation RMSE and test RMSE with these settings. Format these results in a table.

Plot the cross-validation prediction curves (merging the predictions from all splits) for the one-dimensional regression dataset `mauna_loa` at several values of  $k$  for the  $\ell_2$  distance metric. In separate figures, plot the prediction on the test set, as well as the cross-validation loss across  $k$  for this model. Discuss your results.

- Q2) 2pts** Test the performance of your k-NN regression algorithm when a k-d tree data structure is used to compute the nearest neighbours<sup>2</sup> for multiple test points simultaneously.

Conduct performance studies by making predictions on the test set of the `rosenbrock` regression dataset with `n_train=5000`. Report the run-time for varying values of `d` in a single plot. Use the  $\ell_2$  distance metric and  $k = 5$ . Comment on the relative performance of the k-d tree algorithm versus the brute-force approach implemented in your answer to Q1. Use the `time.time` function to measure elapsed wall-clock time for your studies.

- Q3) 2pts** Implement the k-NN algorithm for classification with two different distance metrics ( $\ell_2$  and  $\ell_1$ ). Estimate  $k$  and the preferred distance metric by maximizing the accuracy (fraction of correct predictions) on the validation split. Briefly describe your search procedure to estimate  $k$  and the distance metric. Compute nearest neighbours using a k-d tree data structure, as you had done in Q2.

Apply your algorithm to all classification datasets. For each dataset, report the estimated value of  $k$  and the preferred distance metric, and report the validation accuracy and test accuracy with these settings. Format these results in a table.

---

<sup>1</sup>Note that `data_utils.load_dataset` returns a training and validation set, however, to perform cross-validation, merge these two sets first, i.e. for the inputs: `x_train = np.vstack([x_valid, x_train])`

<sup>2</sup>We do not expect you to implement this data structure. Instead, you may use the scikit-learn implementation `sklearn.neighbors.KDTree` with the default parameters.

- Q4) 4pts** Implement a linear regression algorithm that minimizes the least-squares loss function (using the singular value decomposition). Apply to all datasets (regression and classification). Use both the training and validation sets to predict on the test set, and format your results in a table (present test RMSE for regression, and test accuracy for classification). Compare the performance of this method to the k-NN algorithm.

**Submission guidelines:** Submit an **electronic copy** of your report (**maximum 10 pages** in at least 10pt font) in **pdf** format and **documented** python scripts. You should include a file named “README” outlining how the scripts should be run. Upload a single **tar** or **zip** file containing all files to Quercus. You are expected to verify the integrity of your **tar/zip** file before uploading. Do not include (or modify) the supplied **\*.npz** data files or the **data\_utils.py** module in your submission. The report must contain

- Objectives of the assignment
- A brief description of the structure of your code, and strategies employed
- Relevant figures, tables, and discussion

Do not use scikit-learn for this assignment, except where explicitly specified. Also, do not use the `scipy.spatial` module in this assignment. The intention is that you implement the simple algorithms required from scratch. Also, for reproducibility, always set a seed for any random number generator used in your code. For example, you can set the seed in numpy using `numpy.random.seed`