# Assignment 1: $k$-nearest neighbors and linear regression

CS5824 / ECE5424 – Fall 2020

Out: September 10, 2020
Due: September 20 (11:59 pm), 2020.

**For section 1 and 2, please submit both PDF and .ipynb file saved through Colab with all of your codes and ouputs on. For section 3, please submit a separate PDF file saved from Latex. Zip all these three files without original data files and submit your zipped file (named as FirstNameLastName_Assignment1.zip) through Canvas. Late submissions will be assigned a penalty of 10% for the first late day, 25% for the second late day and will receive a grade of zero on the third late day.**

**Be sure to include your name and student number with your assignment.**

1. **[30 pts]** Classification. Implement $k$-nearest neighbors using Python. Do not use any library such as scikit-learn that already has $k$-nearest neighbor or cross validation implemented. Implementing $k$-nearest neighbor and cross validation from scratch will be a good exercise to make sure that you fully understand those algorithms. Feel free to use general libraries for array and matrix operations such as numpy. Feel free to verify the correctness of your implementation with existing libraries such as scikit-learn.

   Download the dataset posted on canvas/assignments. Classify each input $x$ according to the most frequent class amongst its $k$ nearest neighbors as measured by the Euclidean distance (L2-norm). Break ties at random. Determine the best number of neighbors $k$ by 10-fold cross validation.

   **Dataset:** Dataset for K-nearest neighbor: is posted on canvas/files/assignments/Assignments1.zip/data/knn-dataset

   Problem: this data is a modified version of the Optical Recognition of Handwritten Digits Dataset from the UCI repository. It contains pre-processed black and white images of the digits 5 and 6. Each attribute indicates how many pixels are black in a patch of 4 x 4 pixels.

   Format: there is one row per image and one column per attribute. The class labels are 5 and 6.

   The training set is already divided into 10 subsets for 10-fold cross validation.

   **What to hand in:**

   - Your code for $k$-nearest neighbors **[10 pts]** and cross validation **[5 pts]**.
   - Find the best $k$ by 10-fold cross validation **[5 pts]**. Draw a graph **[5 pts]** that shows the cross validation accuracy as $k$ increases from 1 to 30.
   - Report the best number of neighbors $k$ and the accuracy **[5 pts]** on the test set of $k$-nearest neighbors with the best $k$.

2. **[30 pts]** Regression. Using Python, implement linear least square regression with the penalty term $0.5\lambda w^T w$. Do not use any library such as scikit-learn that already has linear regression or cross validation implemented. Implementing linear regression and cross validation from scratch will be a good exercise to make sure that you fully understand those algorithms. Feel free to use general libraries for array and matrix operations such as numpy. Feel free to verify the correctness of your implementation with existing libraries such as scikit-learn.

Download the dataset posted on the course web page. The output space is continuous (i.e., $y \in \Re$). Determine the best $\lambda$ by 10-fold cross validation.

**Dataset:** Dataset for linear regression: canvas/files/assignments/Assignment1.zip/data/regression-dataset Problem: this data corresponds to samples from a 2D surface that you can plot to visualize how linear regression is working. Format: there is one row per data instance and one column per attribute. The targets are real values. The training set is already divided into 10 subsets for 10-fold cross validation.

**What to hand in:**

- Your code for linear regression **[10 pts]** and cross validation **[5 pts]**.
- Find the best $\lambda$ by 10-fold cross validation **[5 pts]**. Draw a graph **[5 pts]** that shows the cross validation accuracy as $\lambda$ increases from 0 to 4 in increments of 0.1.
- Report the best $\lambda$ and the accuracy **[5 pts]** of linear regression on the test set for this best $\lambda$.

3. **[40 pts]** Theory. There are several possible loss functions for linear regression. However some loss functions assume that the error contributed by each data point has the same importance. Consider a scenario where we would like to give more weight to some data points. Our goal is to fit the data points $(x_n, y_n)$ in proportion to their weights $r_n$ by minimizing the following objective:

$$L(w, b) = \sum_{n=1}^{m} r_n (y_n - w x_n + b)^2$$

where $w$ and $b$ are the model parameters, the training data pairs are $(x_n, y_n)$. To simplify things, feel free to consider 1D data (i.e., $x_n$ and $w$ are scalars).

(a) **[20 pts]** Derive a closed-form expression for the estimates of $w$ and $b$ that minimize the objective. Show the steps along the way, not just the final estimates.

(b) **[20 pts]** Show that this objective is equivalent to the negative log-likelihood for linear regression where each data point may have a different Gaussian measurement noise. What is the variance of each measurement noise in this model?

**Instructions for using Colab:**
Make sure to read submission guidelines from Colab shared example first. Please first download the datasets from Canvas and upload them to google drive. For setting up environment in Colab, please first mount your google drive in Google Colab and append the directory to your python path using sys. We show a simple example about how to set up the environment and how to load data on Colab via:
https://drive.google.com/file/d/1S6uXPuDKoqam9DYeHvZN4KGEEVGx8GMi/view?usp=sharing