Homework 3

Please scan and upload your assignments on or before March 5, 2020.

- You are encouraged to discuss ideas with each other; but
- you **must acknowledge** your collaborator, and
- you **must compose your own** writeup and/or code independently.
- We **strongly** encourage answers to theory questions in Latex, and answers to coding questions in Python (Jupyter notebooks).
- Please upload your solutions in the form of a single .pdf or .zip file on NYUClasses.
- Maximum score: 50 points.

---

1. **(10 points)** Suppose we wish to learn a regularized least squares model:

$$L(w) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \langle w, x_i \rangle)^2 + \lambda R(w)$$

where $R(w)$ is a regularization function to be determined. Suggest good choices for $R(w)$ if the following criteria need to be achieved (there are no unique correct answers) and justify your choice in a sentence or two:

   a. All parameters w are free to be determined.

   b. w should be sparse (i.e., only a few coefficients of w are nonzero).

   c. The coefficients of w should be small in magnitude on average.

   d. For most indices j, w_j should be equal to w_{j-1}.

   e. w should have no negative-valued coefficients.

2. **(10 points)** The *Boston Housing Dataset* has been collected by the US Census Service and consists of 14 urban quality-of-life variables, with the last one being the median house price for a given town. Code for loading the dataset is provided at the end of this assignment. Implement a linear regression model with ridge regression that predicts median house prices from the other variables. Use 10-fold cross validation on 80-20 train-test splits and report the final R^2 values that you discovered. (You may want to preprocess your data to the range $[0, 1]$ in order to get meaningful results.)

3. **(10 points)** In class, we discussed the *lasso* objective, where the regularizer was chosen to be the ℓ_1-norm. Here, we will derive an analytical closed form expression for the minimizer of a slightly simpler problem. Suppose x is a d-dimensional input and w is a d-dimensional variable. Show that the minimizer of the loss function:

$$L(w) = \frac{1}{2}\|x - w\|_2^2 + \lambda \|w\|_1$$

is given by:

$$w_i^* = \begin{cases} x_i - \lambda & \text{if} \quad x_i > \lambda, \\ x_i + \lambda & \text{if} \quad x_i < -\lambda, \\ 0 & \text{otherwise.} \end{cases}$$

4. **(20 points)** In this problem, we will implement logistic regression trained with GD/SGD and validate on synthetic training data.

   a. Suppose that the data dimension d equals 2. Generate two clusters of data points with 100 points each (so that the total data size is $n = 200$), by sampling from Gaussian distributions centered at $(0.5, 0.5)$ and $(-0.5, -0.5)$. Call the data points x_i, and label them as $y_i = \pm 1$ depending on which cluster they originated from. Choose the variance of the Gaussian to be small enough so that the data points are sufficiently well separated. Plot the data points on the 2D plane to confirm that this is the case.

   b. (Derive your own GD routines; do *not* use sklearn functions here.) Train a logistic regression model that tries to minimize:

$$L(w) = -\sum_{i=1}^{n} y_i \log \frac{1}{1 + e^{-\langle w, x_i \rangle}} + (1 - y_i) \log \frac{e^{-\langle w, x_i \rangle}}{1 + e^{-\langle w, x_i \rangle}}$$

   using Gradient Descent (GD). Plot the decay of the training loss function as a function of number of iterations.

   c. Train the same logistic regression model, but this time using Stochastic Gradient Descent (SGD). Demonstrate that SGD exhibits a slower rate of convergence than GD, but is faster per-iteration, and does not suffer in terms of final quality. You may have to play around a bit with the step-size parameters as well as mini-batch sizes to get reasonable answers.

   d. Overlay the original plot of data points on the 2D data plane with the two (final) models that you obtained above in parts b and c to visualize correctness of your implementation.

5. **(optional)** How much time (in hours) did you spend working on this assignment?

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_boston

boston_dataset = load_boston()
boston = pd.DataFrame(boston_dataset.data,
  columns=boston_dataset.feature_names)
boston['MEDV'] = boston_dataset.target
boston.head()
```