EECE5644 Spring 2024 – Homework 3

Submit: Please submit your solutions in a single PDF file using the Canvas/Assignments page. The PDF should include all math, numerical, and visual results. Code can be appended in the PDF or a link to your online code repository can be included. If you point to an online repository, please do not edit the contents after the deadline, because graders may interpret a last-modified timestamp past the deadline as a late submission of the assignment. Only the contents of the PDF will be graded. Please do NOT link from the pdf to external documents online where results may be presented. Any material presented outside the PDF will not be considered for grading. You can use and modify code samples provided along with this handout as you wish to implement the final code needed for the homework.

This is a graded assignment and the entirety of your submission must contain only your own work. You may benefit from publicly available literature including software (not from classmates), as long as these sources are properly acknowledged in your submission. Copying math or code from each other is not allowed and will be considered as academic dishonesty. While there cannot be any written material exchange between classmates, verbal discussions to help each other are acceptable. Discussing with the instructor, the teaching assistant, and classmates at open office periods to get clarification or to eliminate doubts are acceptable.

By submitting a PDF file in response to this take home assignment you are declaring that the contents of your submission, and the associated code is your own work, except as noted in your citations to resources.

Question 1 (15%)

For this question, you should download file `generate_data.py`, read through it and and try to run different parts of the code.

1. Provide the mathematical formula for function `fun`. What coordinates of vector $x \in \mathbb{R}^5$ does its output depend on? Is it a linear function of $x \in \mathbb{R}^5$?

2. Describe, in your own words, what the code in `generate_data.py` does.

3. What happens when you change variable N to $N = 2000$?

4. What happens when you set variable d to $d = 10$? Does function `fun` still work? What coordinates of vector $x \in \mathbb{R}^{10}$ does its output depend on?

Question 2 (25%)

For this question, you should download and modify file `LinearRegression.py`.

1. The Root Mean Square Error (RMSE) between a vector of ground-truth variables $y \in \mathbb{R}^n$ and a vector of corresponding predictions $\hat{y} \in \mathbb{R}^n$ is given by

$$\sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}.$$

   Explain how this relates to the training objective of linear regression, i.e., the function linear regression minimizes to obtain the least-squares estimate of a linear model.

2. Read the documentation of the `sklearn.linear_model.LinearRegression` class and describe (a) what methods:

   `fit`, `predict`

   do, and (b) what attributes:

   `coef_`, `intercept_`

   mean/what information they contain.

3. Copy file `LinearRegression.py` and modify it, to create a new script called `LinearRegressionSweepN.py` that does the following:

   - It reads a dataset generated by `generate_data.py`.
   - Splits this into a training and a test set, with a %70/%30 split ratio.
   - Trains multiple ten different linear regression models, *using only a fraction* `fr` of the training set each time, where

     `fr`=10%, 20%, ..., 100%.

     For each fraction value, the code should compute the RMSE of predictions both on the training samples you used to train the model, as well as the RMSE on the *entire* test set.

Run your code on a dataset generated by `generate_data.py` with $N = 1000$, $d = 5$, and $\sigma = 0.01$. Plot the train and test RMSE as a function of the number of training samples given to your model. Does increasing the number of samples significantly reduce either errors? Do you think that increasing the number of samples beyond 1000 would help in significantly lowering either errors? If not, why?

4. Run your code on a dataset generated by `generate_data.py` with $N = 1000$, $d = 40$, and $\sigma = 0.01$. Note that, by increasing the size of the input vector, you are in fact adding variables that do not affect your labels; these are sometimes called *nuisance variables*. For this dataset, plot the train and test RMSE as a function of the number of training samples given to your model. Do you see any differences from Q2.3? Explain what you think is happening to cause this behavior. Do you think that increasing the number of samples beyond 1000 would significantly reduce the test error?

Question 3 (35%)

1. Define a python function called `lift` that takes as input a vector $x \in \mathbb{R}^d$ and outputs a vector $x \in \mathbb{R}^{d'}$ that contains x as well as all possible coordinate combinations of the form $x_i \cdot x_j$, for $i, j \in \{1, \ldots, d\}$, with $i \geq j$. For example, vector

$$x = [x_0, x_1, x_2, x_3, x_4] \in \mathbb{R}^5$$

should be converted to

$$\begin{aligned} x' = \big[&x_0, x_1, x_2, x_3, x_4, \\ &x_0^2, x_1x_0, x_1^2, x_2x_0, x_2x_1, x_2^2, \\ &x_3x_0, x_3x_1, x_3x_2, x_3^2, x_4x_0, x_4x_1, x_4x_2, x_4x_3, x_4^2\big] \in \mathbb{R}^{20} \end{aligned}$$

Your code should work for any $d \in \mathbb{N}$. Add the code in your report.

2. Show that function `fun` can be written as a linear function of such a lifted vector x', i.e., a function of the form:

$$f(x') = \beta^\top x' + c,$$

for an appropriately defined vector $\beta \in \mathbb{R}^{d'}$ and constant $c \in \mathbb{R}$, and provide β and c.

3. Define a python function `liftDataset` that takes as input a matrix $X \in \mathbb{R}^{n \times d}$ and produces a matrix $X' \in \mathbb{R}^{n \times d'}$ that contains all rows of X, but lifted. You can use function `lift` to implement this. Add the code in your report.

4. Modify `LinearRegressionSweepN` so that it lifts matrix X generated by `generate_data` right after reading it. This should happen *before* splitting the dataset into a train and test set. Run your code on a dataset generated by `generate_data.py` with $N = 1000$, $d = 5$, and $\sigma = 0.01$. Plot the train and test RMSE as a function of the number of training samples given to your model. How does this plot differ from previous plots you generated? Why? Obtain the parameters (coefficients and intercept) of your finally trained model. Did you manage to learn function `fun`?

5. Repeat this now on a dataset (also lifted) generated by `generate_data.py` with $N = 1000$, $d = 40$, and $\sigma = 0.01$. How does the training and test error change as you increase the number of training samples? Obtain the parameters (coefficients and intercept) of your finally trained model. Did you manage to learn function `fun`? Do you think that increasing the number of samples beyond 1000 would improve the test error/the proximity of your learned model to the true function `fun`?

Question 4 (25%)

In this question, you will try to learn the correct coefficients for the model function `fun` even when $d = 40$, *without increasing the number of samples.*

1. Read through the code in file `Lasso.py` and explain what it does.

2. Create a file called `LassoSweepAlpha.py` that is a modified version of Lasso and implements the following functionality:

   - First, it lifts the dataset before splitting it to a train and test set.
   - Then, it performs 5-fold cross validation on the training set for different Lasso regularization parameters α, ranging from 2^{-10} to 2^{10}, and finds the α that minimizes the mean RMSE across folds.
   - Finally, the code should re-train a Lasso model over *the entire training set* under the optimal value of α, and compute the train and test RMSE.

   Run your code on a dataset generated by `generate_data.py` with $N = 1000$, $d = 40$, and $\sigma = 0.01$. Plot the cross-validation mean RMSE as a function of α (include error bars).

3. Report also the train and test RMSE for the model with an optimal α. Report also all the parameters (coefficients and intercept) of your finally trained model that have an absolute value larger than 10^{-3}. Did you do better at learning function `fun` compared to Q3.5, despite all the nuisance variables? If so, why?