CSI 5340 Homework Exercise I

This homework is programming based, where **you must use Python** to explore the fitting and generalization of regression models via simulation.

Suppose that X and Y are both real valued random variables, where X takes value in $(0, 1)$ and Y depends on X according to

$$Y = \cos(2\pi X) + Z \tag{1}$$

where Z is a zero mean Gaussian random variable with variance σ^2, and Z is independent of X. But assume that you do not know this dependency of Y on X and that you only observe a sample of N (X, Y) pairs. Based on the observed sample, you must learn a polynomial regression model and examine the fitting and generalization capability of your model in relation to the model complexity and sample size.

Below detailed instructions are given to guide you through this exercise. These instructions only serve as a guideline, which your implementation need not to rigorously follow. You must use Python to write your code. It is fine and encouraged, but NOT compulsory, if you use a Python package that does automatic differentiation.[1] **But you must implement manually gradient-based optimization. That is, the use of the package is only for you to compute the required gradients and you must manually code up the update of model parameters.** You need to submit the following deliverables in a single zipped file.

- All Python code
- A report explaining your findings.

Note that the exercise is meant to be a careful analysis of this regression problem with and without weight-decay regularization. **Take it as a mini-research paper.** That is, your report is not only graded based on its correctness and whether it has included minimally required experimental results. The following factors are also considered in grading.

- Understanding of fitting and generalization in relation to sample size and model complexity, and impact of noise
- Interesting observations, explanations, and insights
- Presentation

(A) Write a function `getData` that generates a dataset $\{(x_i, y_i) : i = 1, 2, \ldots N\}$ of N (X, Y) pairs for a given value of N and σ^2. The X values are drawn uniformly at random from $(0, 1)$ and the corresponding Y values are generated according to (1).

The dataset created by `getData` will then be used to fit your regression models. Of course, in the design of your regression model, you should assume no knowledge on how the dataset is generated.

The regression models we consider will be exclusively polynomial models, namely, predicts Y from X according to

$$Y = a_0 + a_1 X + a_2 X^2 + \ldots + a_d X^d$$

[1]If you are new to such packages, I recommend PyTorch.

where d is the polynomial degree and a_i's are coefficients to be estimated.

(B) Write a function `getMSE` which computes the mean square error (MSE) for a given dataset fitted to a specified polynomial.

(C) Write a function `fitData` that estimates the polynomial coefficients by fitting a given dataset to a degree-d polynomial. The function returns the following:

1. The estimated polynomial coefficients. The estimation of the coefficients should be based on GD.
2. The MSE of the dataset fitted to the estimated polynomial. This MSE will be denoted by E_{in}.
3. E_{out}. To obtain this value, your function needs to generate a separate large testing dataset (say, containing 1000 or 2000 data points) using `getData` and under the same setting of σ^2) and compute the MSE of the testing dataset fitted to the estimated polynomial.

The computation of E_{in} and E_{out} calls `getMSE`.

(D) Write a function `experiment` that takes as input the size N of training dataset, the degree d of the model polynomial and noise variance σ^2, and does the following. For the given values of N,d and σ^2, it loops over M trials (M not smaller than 20; say, 50 would be a decent number), where each trial is defined as generating a training dataset of size N and noise variance σ^2 (by calling `getData`) and then fitting the data to a polynomial of degree d (by calling `fitData`). The computed E_{in} and E_{out} are respectively averaged over the M trials, which are denoted by $\overline{E}_{\text{in}}$ and $\overline{E}_{\text{out}}$. For each run of `experiment` with input configuration (N, d, σ^2), the program outputs $(\overline{E}_{\text{in}}, \overline{E}_{\text{out}})$.

(E) Run `experiment` for all combinations of N, d, and σ^2, with $N \in \{2, 5, 10, 20, 50, 100, 200\}$, $d \in \{1, 2, 4, 8, 16, 32, 64\}$, $\sigma \in \{0.05, 0.2\}$. Observe the behaviour of $(\overline{E}_{\text{in}}, \overline{E}_{\text{out}})$ in relation to model complexity (d), sample size (N), and noise level (σ). Then make the following plots, each containing four curves: $\overline{E}_{\text{in}}$ for $\sigma = 0.05$, $\overline{E}_{\text{out}}$ for $\sigma = 0.05$, $\overline{E}_{\text{in}}$ for $\sigma = 0.2$, $\overline{E}_{\text{out}}$ for $\sigma = 0.2$.

1. Pick three values of N, and for each picked value, create a plot of the four errors as functions of d.
2. Pick three values of d, and for each picked value, create a plot of the four errors as functions of N.

Discuss your observations from the six plots. Feel free to include and discuss additional results (which may be presented in other format).

(F) Revise your code to include weight decay regularization, and redo **(E)**. Discuss your observations, particularly in comparison with the observations in **(E)**. Make additional plots if needed to illustrate the comparison.