ENSAI **Machine learning** **Computer class**

This is personal work which will take part in your final evaluation: 50 percent of your final grade (the other 50 percent are given by the final exam). In the end each student shall provide a single `ipynb` file containing his or her answers to the questions. The notebook must be sent to by email me using `francois.portier@ensai.fr` before **Sunday 2th of November, 23:59**. Out of 20 points, 5 are specifically dedicated to:

- Presentation quality: writing, clarity, no typos, visual efforts for graphs, titles, legend, colorblindness, etc. (2 points).
- Coding quality: indentation, PEP8 Style, readability, adapted comments, brevity (2 points)
- No bug on the grader's machine (1 point). Latest version of python is recommended but any older version of `python3` should do the job.

You can use `https://www.python.org/dev/peps/pep-0008/` to get some info about PEP8. In addition, some of the questions are kind of "open" leaving some room to the students personal developments. The originality of the given answer will be valued.

(Q1) In the (random design) linear regression framework where $Y = Z\theta^* + \epsilon$ (see the course for the introduction of the precise notation), the expected excess risk bound was shown to scale as $\mathbb{E}[R(\hat{\theta})] - R(\theta^*) = \sigma^2(p+1)/n$ where σ^2 is the variance noise, n is the number of observation and p is the number of covariates. Using some computer experiments and considering a simulation set-up with independent Gaussian covariates $X_i \sim \mathcal{N}(0, I_p)$ where I_p is identity matrix of size p and Gaussian noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, illustrate the previous statement (here the covariates are random). Formulas from the course should be recalled and explained with care to make the illustration clear and sound.

(Q2) From now on we consider the following data generation process. Let $p = 8$ and let X_i be a Gaussian vector as before. For each $i = 1, \ldots, n$,

$$Y_i = X_{i1} + (X_{i2})_+ + \epsilon_i.$$

where ϵ_i is a Gaussian noise $\mathcal{N}(0, \sigma^2 = .5)$ and $(x_+)+ = x1_{(0,+\infty)}(x)$. Generate an independent and identically distributed collection of $n = 3000$ random variables from this distribution. With the help of simple graphs show that x_1 and x_2 have a particular relationship with the output y compared to the others.

(Q3) Define the linear regression estimator as

$$\hat{\beta}_n \in \arg\min_{\beta \in \mathbb{R}^{p+1}} \{f(\theta) := n^{-1}\|Y - X\beta\|_2^2\}.$$

Solve the above using standard implementation from sklearn. Print the empirical risk's value for different values of $n \in \{100, 500, 1000, ..., 3000\}$. Give some comments about what is observed.

(Q4) (first-order) No package should be used here except the basics like *numpy*. Apply the gradient descent (GD) algorithm algorithm to the previous optimisation problem. The function to optimize can be written as

$$f(\theta) = n^{-1} \sum_{i=1}^{n} f_i(\theta)$$

where $f_i(\theta) = (Y_i - X_i^T\theta)^2$. Apply the stochastic gradient descent (SGD) algorithm to the above while sampling uniformly among the 3000 coordinates. In both GD and SGD, learning rate will be taken as $1/k$, k being the iteration of the optimization algorithm. The number of evaluation of the gradient of $f_i(\theta)$ shall be the same for each algorithm (i.e., take care that one step of GD takes many more evaluations). Provide pseudo-code to explain both algorithms. With the help of several experimental runs, and plotting the decrease of the function value f, visualize the randomness (and non-randomness) of SGD (of GD).

(Q5) (zero-order) Now is supposed that we cannot evaluate the gradient. Explain that, when h is small,

$$(2h)^{-1}U\{f(\theta + hU) - f(\theta - hU)\} \simeq UU^T\nabla f(\theta)$$

Propose a way to generate U in the perspective gradient descent algorithms. Write a pseudo-code to describe an algorithm relying on the above estimate of gradient. The first proposed algorithm, called ZGD, should use full function evaluation $f(\theta) = n^{-1}\sum_{i=1}^{n} f_i(\theta)$ at each iteration (i.e., when estimating the gradient), but a stochastic version, called ZSGD, should also be provided. Compare the two algorithms to the standard GD and SGD from before.

(Q6) Now consider a more flexible model defined as

$$f_{\beta,\gamma}(x) = x^T\beta + (x^T\gamma)_+. \tag{1}$$

Train the model with ZSGD and ZGD and compare them in terms of the value function f while using the same number of evaluation of f for both methods. Evaluate/Estimate the risk using $n = 1000$ additional generations from the previous data generation process. Compare the two above approaches: (a) ZSGD with the complex model and (b) ZSGD with the linear model (with fixed number of evaluation of $f_i(\beta)$).

(Q7) (bonus question) Consider the classification data from `https://archive.ics.uci.edu/dataset/159/magic+gamma+telescope`. Train one base line model e.g., logistic regression, and compare it to your own model which might be anything (e.g., you can still do logistic regression or neural network classification) but training should be based on zero-th stochastic gradient descent (as before). Explain and give the details about the procedure you propose.