The Chinese University of Hong Kong, Shenzhen
School of Data Science

DDA5001 · Homework 2
Due: (23:59), Oct 26.

Instructions:

- Homework problems must be clearly answered to receive full credit. Complete sentences that establish a clear logical progress are highly recommended.
- Submit your answer paper in Blackboard. Prepare your answers in a **PDF** file and saving all your code as **.py** files. Combine the PDF and code files into a single .zip file for submission. The file name should be in the format **last name-first name-hw2**.
- The homework must be written in English.
- Late submission will not be graded.
- Each student **must not copy** homework solutions from another student or from any other source. You are encouraged to discuss with others. However, you must write down the answer using your understanding (after discussion) and words.

Problem 1 (16pts). Concepts and Fundamental Knowledge

(a) For a d-dimensional binary linear classifier, what is its VC dimension?

(b) Briefly state the difference between test error and out-of-sample error.

(c) To make out-of-sample error small, we only need to make in-sample error as small as possible [true/false]. If your answer is false, briefly state the reason.

(d) Which one of the following statements is true? Briefly state the reasons why the other three choices are wrong.

1) Logistic regression (LR) is designed for linear regression.
2) Closed-form solution exists for LR.
3) LR is a linear classifier.
4) LR can only be applied to binary classification.

Problem 2 (25pts). Multi-class Logistic Regression

In this problem, we will explore the performance of multi-class logistic regression under different optimization methods and different datasets.

(a) Let $\boldsymbol{\Theta} \in \mathbb{R}^{K\times d}$. We have learned that the a-posterior of y_i can be modeled as

$$\Pr_{\boldsymbol{\Theta}}[y_i = \ell|\boldsymbol{x}_i] = \frac{\exp((\boldsymbol{\theta}^\ell)^\top \boldsymbol{x}_i)}{\sum_{j=1}^K \exp((\boldsymbol{\theta}^j)^\top \boldsymbol{x}_i)}.$$

We can add bias terms $\boldsymbol{b} \in \mathbb{R}^K$ to enable better expressivity of the model (In lecture, we absorbed the bias terms $\boldsymbol{b}$ in $\boldsymbol{\Theta}$). In this case, the a-posterior of y_i can be modeled as

$$\Pr_{\boldsymbol{\Theta},\boldsymbol{b}}[y_i = \ell|\boldsymbol{x}_i] = \frac{\exp((\boldsymbol{\theta}^\ell)^\top \boldsymbol{x}_i + \boldsymbol{b}^\ell)}{\sum_{j=1}^K \exp((\boldsymbol{\theta}^j)^\top \boldsymbol{x}_i + \boldsymbol{b}^j)}.$$

Using MLE, the learning problem can be formulated as

$$\min_{\boldsymbol{\Theta},\boldsymbol{b}} \ \mathcal{L}(\boldsymbol{\Theta},\boldsymbol{b}) := -\frac{1}{n}\sum_{i=1}^n \sum_{\ell=1}^K \mathbf{1}_{\{y_i=\ell\}} \log\left(\frac{\exp((\boldsymbol{\theta}^\ell)^\top \boldsymbol{x}_i + \boldsymbol{b}^\ell)}{\sum_{j=1}^K \exp((\boldsymbol{\theta}^j)^\top \boldsymbol{x}_i + \boldsymbol{b}^j)}\right). \tag{1}$$

Derive the gradient $\frac{\partial \mathcal{L}(\boldsymbol{\Theta},\boldsymbol{b})}{\partial \boldsymbol{\Theta}}$ and $\frac{\partial \mathcal{L}(\boldsymbol{\Theta},\boldsymbol{b})}{\partial \boldsymbol{b}}$. We have provided the essential derivations on LR's gradient calculation, along with the announcement of HW2 called "Note_matrix_differentiation". You may refer to page 7-8 in the this supplementary material, and then you only need to derive a small step further.

(b) We will experiment the algorithm on two datasets. The first dataset consists of coffee bean images from 4 categories: Dark, Green, Light, and Medium. The second dataset contains pictures taken from 5 different weathers: Cloudy, Foggy, Rainy, Shine, and Sunrise. In this question, we will perform classification using logistic regression. To optimize the objective (1), we will explore both the gradient descent method and the accelerated gradient descent method. The detailed tasks are:

- For each dataset, plot the training loss and test loss of two optimization algorithms in the same figure (4 curves in the same figure).
- For each dataset, plot the training accuracy and test accuracy of two optimization algorithms in the same figure (4 curves in the same figure).
- Briefly summarize your observations for two algorithms' performance on these two datasets in the following perspectives: Convergence speed, accuracy, overfitting.

(a) Dark (b) Green (c) Light (d) Medium

Figure 1: Different categories of coffee bean.

(a) Sunrise

(b) Cloudy

Figure 2: Different weathers.

The uploaded source code contains the data-preprocessing and train-test splitting, which should not be changed. The parts marked by `#TODO` should be completed. The suggested parameters are total iteration $T = 1000$, learning rate $\mu = 0.01$ for the coffee bean dataset, and $\mu = 0.02$ for the weather dataset. **Note: Do not use any pre-built packages for auto-differentiation.**

Problem 3 (14pts). Subgradient Method for Optimizing Non-Smooth Function

The smoothness property ensures that the gradient exists, so that we can apply the gradient descent method to optimize the function. When optimizing non-smooth functions such as ℓ_1-norm, we have shown in homework 1 that we can transform the ℓ_1 norm into a smooth objective by using "Huber smoothing" technique, and then use gradient descent to optimize it.

In this problem, we introduce another approach for optimizing non-smooth function directly, namely, the *subgradient method*. We will study the subgradient method in detail later in our lecture. We say a vector $\boldsymbol{g} \in \mathbb{R}^n$ is a *subgradient* of $f : \mathbb{R}^n \to \mathbb{R}$ at $\boldsymbol{x} \in \mathbb{R}^n$ if for all $\boldsymbol{z} \in \mathbb{R}^n$:

$$f(\boldsymbol{z}) \geq f(\boldsymbol{x}) + \boldsymbol{g}^\top (\boldsymbol{z} - \boldsymbol{x}).$$

The collection of all subgradients is called *subdifferential*, denoted by $\partial f(\boldsymbol{x})$. If f is differentiable, then its subdifferential is a singleton and coincides with the gradient. When f is not differentiable at $\boldsymbol{x}$, there may exist more than one subgradient, as shown in Figure 3.

The subgradient method recursively performs the following update:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \mu_k \boldsymbol{g}_k, \quad \boldsymbol{g}_k \in \partial f(\boldsymbol{x}_k),$$

where μ_k is the learning rate and $\boldsymbol{g}_k$ is any subgradient at $\boldsymbol{x}_k$.

We now consider using the subgradient method to directly optimize the robust linear regression learning problem in homework 1:

$$\min_{\boldsymbol{x}} \ \mathcal{L}(\boldsymbol{x}) := \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_1. \tag{2}$$

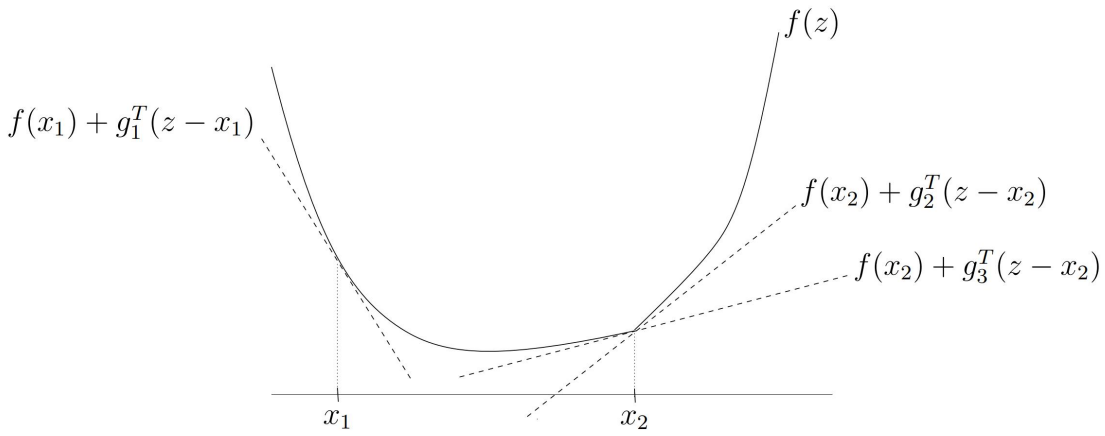

Figure 3: At $\boldsymbol{x}_1$, the convex function f is differentiable, and $\boldsymbol{g}_1$ (which is the gradient of f at $\boldsymbol{x}_1$) is the unique subgradient at $\boldsymbol{x}_1$. At the point $\boldsymbol{x}_2$, f is not differentiable. Hence, f has infinitely many subgradients at this point: two subgradients, $\boldsymbol{g}_2$ and $\boldsymbol{g}_3$ are shown.

For the objective function f in (2), we can apply chain rule to compute its subdifferential:

$$\partial \mathcal{L}(\boldsymbol{x}) = \boldsymbol{A}^\top \operatorname{sign}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}).$$

Here, $\operatorname{sign}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$ can be a set since $\operatorname{sign}(0) = [-1, 1]$.

(a) Load the data `"A.npy"` and `"b.npy"`, where $\boldsymbol{A} \in \mathbb{R}^{n \times d}$, $\boldsymbol{b} \in \mathbb{R}^n$ with $n = 500$, $d = 30$. Solve problem (2) by implementing the subgradient method with three learning rate strategies:

- Constant learning rate: $\mu_k = \mu = 0.005$.
- Polynomial diminishing learning rate: $\mu_k = \frac{0.03}{\sqrt{k}}$.
- Geometrically diminishing learning rate: $\mu_k = 0.01 * 0.9^k$.

Set the total iteration number to 300. Load the optimal solution $\boldsymbol{x}^*$ from `"p4_x_star.npy"`. Plot the optimality gap $\|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2$ obtained by the three learning rate schedules **twice**: 1) one figure uses normal scale for x-axis and y-axis, and 2) another figure uses normal scale for x-axis while uses log-scale for y-axis. Briefly discuss your observations in the following perspectives: Whether converge to optimal solution? What are the speed of each learning rate schedule? Any difference from the gradient descent method?

Problem 4 (45pts). Overfitting

This problem aims to study the overfitting issue through an experiment design.

Legendre polynomial. We first introduce Legendre polynomial, which serves as the preliminary of our experiment setting. The first two Legendre polynomials are $L_0(x) = 1, L_1(x) = x$. The higher-order Legendre polynomials are defined recursively as

$$L_k(x) = \frac{2k-1}{k} x L_{k-1}(x) - \frac{k-1}{k} L_{k-2}(x).$$

It is easy to see that L_k is a polynomial function of degree k. That is, L_k can be written equivalently as $L_k(x) = c_0 + c_1 x + \ldots + c_q x^q + \ldots + c_k x^k$ for some proper coefficients $\{c_q\}_{q=0}^k$.

Legendre polynomial of different orders are orthogonal to each other and the square of each of them has a finite integral within the interval $[-1, 1]$, i.e.,

$$\int_{-1}^{1} L_q(x) L_i(x) dx = \begin{cases} 0 & i \neq q, \\ \frac{2}{2q+1} & i = q. \end{cases}$$

Underlying target function. Consider the underlying polynomial target function

$$g(x) = \frac{1}{C_Q} \sum_{q=0}^{Q_g} a_q L_q(x), \quad \text{with } x \text{ sampled from } [-1, 1] \text{ uniformly at random.}$$

Here, Q_g is the maximum degree of the Legendre polynomial, each coefficient $a_q \sim \mathcal{N}(0, 1)$ is a random variable and follows a standard Gaussian distribution, L_q is the Legendre polynomial with degree q, and $C_Q = \sqrt{\sum_{i=0}^{Q_g} \frac{1}{2i+1}}$. C_Q is used to normalize the variance, giving that $\mathbb{E}_x[g^2(x)] = 1$, where the expectation is taken over the randomness of x.

Generating training dataset. We can generate the training dataset $\mathcal{D} = \{(x_1, y_1), \cdots, (x_n, y_n)\}$, with each x_i being independently sampled from $[-1, 1]$ uniformly at random and $y_i = g(x_i) + \sigma \epsilon_i$, where g is the target function defined above and ϵ_i is i.i.d. standard Gaussian random noise.

Problem setting. We study how the noise level σ^2, the target complexity Q_g, and the number of data points n affect overfitting. Towards this end, we will choose the hypothesis space to be polynomial functions generated by Legendre polynomials, i.e., we define the model

$$f_K(x) = \sum_{k=0}^{K} w_k L_k(x),$$

where K is the maximum degree of the Legendre polynomial and $\{w_k\}$ are the trainable parameters. Our goal is to learn a f_K (i.e., $\{w_k\}$) to approximate g based on the training dataset $\mathcal{D}$ generated above. The in-sample error and out-of-sample error of f are defined as

$$\mathrm{Er}_{\mathrm{in}}(f) = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)^2, \quad \mathrm{Er}_{\mathrm{out}}(f) = \mathbb{E}_x \left[(f(x) - g(x))^2 \right].$$

Let f_2 and f_{10} be the best fit (i.e., lowest $\mathrm{Er}_{\mathrm{in}}$) with degree of 2 and 10, respectively. We study the overfitting behavior of f_{10} under different conditions. To provide a quantity for measuring overfitting, we use the overfitting measure defined as

$$\varrho_e := \mathrm{Er}_{\mathrm{out}}(f_{10}) - \mathrm{Er}_{\mathrm{out}}(f_2).$$

If ϱ_e is significantly positive, we have that f_{10} must be overfitting (f_2 may be underfit, goodfit, or overfit). If ϱ_e is clearly negative, we have that f_{10} must not be overfitting (f_{10} may be underfit or goodfit, while f_2 must be be underfit). Think why we have these claims.

(a) Suppose that $n \geq K+1$. How can we obtain f_2 and f_{10} by minimizing the corresponding $\mathrm{Er}_{\mathrm{in}}$? (**Hint:** This is just a least squares problem for learning $\{w_k\}$ based on the training dataset. So, constructing the data matrix and label vector in least squares gives the result immediately.)

(b) How can we compute $\mathrm{Er}_{\mathrm{out}}$ analytically for the learned f_2 and f_{10} in part (a)? (**Hint:** Write down the definition of $\mathrm{Er}_{\mathrm{out}}$ and then proceed by utilizing the orthogonality of Legendre polynomials.)

(c) Since we have investigated how to calculate $\mathrm{Er}_{\mathrm{out}}$ analytically in part (b), now we can conduct experiments to illustrate the behavior of the overfitting measure ϱ_e using different choices of Q_g, n, σ^2. Specifically, complete the following tasks:

1) Fix $\sigma^2 = 0.1$, while vary $Q_g \in \{1, 2, \cdots, 50\}$ and $n \in \{20, 25, \cdots, 120\}$. For each pair of (n, Q_g), use the provided code `p4.py` to generate $\{a_q\}_{q=0}^{Q_g}$ (hence the target function g), the samples $\{x_i\}_{i=1}^{n}$, and the labels $\{y_i\}_{i=1}^{n}$. Then, use the result in part (a) to find the optimal coefficients $\{w_k\}$ for both f_2 and f_{10}. Furthermore, use the result in part (b) to finally compute $\mathrm{Er}_{\mathrm{out}}(f_2)$ and $\mathrm{Er}_{\mathrm{out}}(f_{10})$, and hence ϱ_e.

   Repeat such experiment for 40 independent trials and compute the averaged value of ϱ_e (denoted as $\bar{\varrho}_e$) out of these 40 realizations. Visualize $\bar{\varrho}_e$ as a heat map for each pair of (n, Q_g) with x-axis being n and y-axis being Q_g using the provided code.

2) Now, we fix $Q_g = 20$, while vary $\sigma^2 \in \{0, 0.05, \cdots, 2\}$ and $n \in \{20, 25, \cdots, 120\}$. Repeat the same procedures as in 1) to compute $\bar{\varrho}_e$ for each pair of (n, σ^2). Then, visualize $\bar{\varrho}_e$ as a heat map for each pair of (n, σ^2) with x-axis being n and y-axis being σ^2 using the provided code.

3) Briefly summarize what you observe from these experiments in the following perspectives: What can be the causing factors of overfitting? And how they affect overfitting?