Homework 1

Requirement

- Some exercises are optional (as marked below); do them for your interest.
- Compile everything (including your derivations, answers, and source codes) to a pdf file. In the main text, provide your name, student ID, and your email address (marked assignment may be sent to your email). Name your pdf file in the form of "`StudentID_name_1st_homework.pdf`".
- Deadline: 23:59, 26 Sep 2025.
- To be submitted individually.
- Submission method will be announced later.

1 (Mostly) Compulsory

Exercise 1:

The Shannon entropy of a distribution $p(x)$ is a fundamental object in information theory. Assume the random variable has K possible states, i.e., $x \in \mathcal{X} = \{1, 2, 3, ..., K\}$, the entropy is defined as

$$H(p) = -\sum_{x\in\mathcal{X}} p(x)\log p(x) = -\sum_{i=1}^{K} p_i \log p_i,$$

where we adopt the convention $0\log 0 = 0$. The Shannon entropy measures the uncertainty of the variable's possible outcomes. The maximum entropy principle offers a method to select a distribution that is compatible with the data, while using the "least" prior assumption (Laplace's principle of insufficient reason).

Consider an observable $f(x)$ on the random variable x. Our device is only able to measure its expectation

$$\sum_{x\in\mathcal{X}} p(x)f(x) = \sum_{i=1}^{K} p_i f_i = F.$$

Based on this, to choose a distribution by using maximum entropy principle, we need to solve the following constrained optimization problem:

$$\begin{aligned} &\max_p \ H(p) = -\sum_{i=1}^{K} p_i \log p_i \quad (\text{or} \quad \min_p -H(p) \ \text{ in standard convex optimization}) \\ &\text{s.t.} \ \sum_{i=1}^{K} p_i = 1, \quad \sum_{i=1}^{K} p_i f_i = F. \end{aligned}$$

(to think about: why we do not explicitly need the constraint $p_i \geq 0$?)

Questions:

1. Derive the Lagrangian dual function of the above optimization problem.
2. Show that the maximum-entropy distribution belongs to the exponential family

$$p_i = \frac{1}{Z_\lambda} \exp(-\lambda f_i),$$

where the normalization factor $Z_\lambda = \sum_k \exp(-\lambda f_k)$, and λ is a parameter determined by the value of F.

Exercise 2:

LASSO regression and compressed sensing can have very similar formulations. For example, both of them can have a basis pursuit denoising formalism,

$$\begin{aligned} \text{LASSO:} \quad & \min_{\beta} \ 1/2\|y - X\beta\|_2^2 + \lambda\|\beta\|_1, \\ \text{Compressed sensing:} \quad & \min_{x} \ 1/2\|y - Ax\|_2^2 + \lambda\|x\|_1. \end{aligned}$$

Describe what's the difference between the two tasks.

Exercise 3:

This is a reading exercise that aims to help you bridge the gap between the theory of compressed sensing and practice.

Read (as much as you like) the tutorial file named "`Weblog_Compressed Sensing in Python.pdf`". You can also run the experiments therein.

Write down what you learned from this tutorial.

An artificial scenario for motivation:

The part "`Reconstruction of a Simple Signal`" in the tutorial may seem quite arbitrary. Below I make up an artificial scenario to give you a better motivation.

Alice is confronted with the task of measuring sound signals from a source. For a time duration T, suppose it is sufficient to record n evenly-spaced data points to represent the analog signal $s(t)$:

$$\{(t, s(t)) \ \mid \ t = i \times \Delta t, \ i = 1, 2, ..., n\},$$

where $\Delta t = T/n$. Denote the n-th dimensional ground truth signal as $\boldsymbol{s} \in \mathbb{R}^n$.

However, Alice's device has too little memory to store all the data points. But luckily, he knew that the sound source generates sounds of only **a few** (unknown) frequencies. Alice decided to use **compressed sensing** to reduce the burden of recording. He randomly sampled a subset of m data points to record: $\{(i, s_i) \mid i \in \mathcal{M}\}$, where $\mathcal{M}$ is the array of the sampled indices and $|\mathcal{M}| = m \ll n$. Based on this information, he wanted to recover the whole signal $\boldsymbol{s}$. Of course, this is only possible if we have the correct assumption of the data (here it is sparsity).

In the language of compressed sensing, the recorded signals $\{s_i \mid i \in \mathcal{M}\}$ are outcome of the measurement $\boldsymbol{y} \in \mathbb{R}^m$

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{s}[\mathcal{M}] = (s_{\mathcal{M}_1}, s_{\mathcal{M}_2}, ..., s_{\mathcal{M}_m}) \\ &:= \mathcal{C} \cdot \boldsymbol{s}. \end{aligned}$$

We have introduced the $m \times n$ sampling matrix $\mathcal{C}$, with $\mathcal{C}_{ij} = 1$ if $j = \mathcal{M}_i$ (otherwise $\mathcal{C}_{ij} = 0$).

This is not yet a compressed sensing problem, as the signal $\boldsymbol{s}$ is not sparse. But we know that the signal is sparse in the **frequency domain**, so we hope to recover the sparse frequency components first. For discretized data such as $\boldsymbol{s}$, a common choice for mapping to the frequency domain is the discrete cosine transform (DCT), represented by a matrix $\Phi \in \mathbb{R}^{n \times n}$. The frequency coefficients of a sound signal $\boldsymbol{s}$ is: $\boldsymbol{x} = \Phi \boldsymbol{s}$. On the other hand, knowing the frequency coefficients $\boldsymbol{x}$, the signal can be found by the inverse DCT: $\boldsymbol{s} = \Phi^{-1}\boldsymbol{x}$. Combining all parts, we have

$$\boldsymbol{y} = A\boldsymbol{x}, \quad \text{with } A = \mathcal{C} \cdot \Phi^{-1}.$$

Applying compressed sensing algorithms to solve the above under-determined linear equations (remind $m < n$) yields an estimate of frequency components $\boldsymbol{x}^*$. The sound signal can then be recovered by inverse DCT: $\boldsymbol{s}^* = \Phi^{-1}\boldsymbol{x}^*$.

In Python, the inverse DCT matrix for a 1D signal can be generated by using the `scipy.fftpack` package:

```
iDCT = scipy.fftpack.idct(np.identity(n), norm='ortho', axis=0)
```

Optional practice

This part is optional. Do it if you find it interesting.

For the above sound signal scenario, consider the extension that the signal is corrupted by noise

$$\boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 I),$$

the reconstruction can be achieved by solving the basis pursuit denoising problem

$$\min_{\boldsymbol{x}} \quad \frac{1}{2} \|\boldsymbol{y} - A\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_1 .$$

Implement the proximal graident algorithm (i.e, the iterative soft-thresholding algorithm) for the signal reconstruction.

2 Optional

Exercises in this section are only optional. Do them if you find them interesting.

Exercise 4:
The "log-sum-exp" function is defined as

$$f(\boldsymbol{x}) = \log \sum_{i=1}^{K} \exp(x_i).$$

It is closely related to the "softmax" function commonly used in classification tasks. In fact, it can make a smooth surrogate of the "max" function

$$\lim_{\beta \to \infty} \frac{1}{\beta} \log \sum_{i=1}^{K} \exp(\beta x_i) \implies \max(x_1, x_2, ..., x_K).$$

Show that the "log-sum-exp" function is a convex function. (See Appendix. A for more details on convexity.)

Exercise 5:

Implicit Bias of Gradient Descent (Exercise 2.10 of [Wright and Ma 2022])

Consider the problem of solving an under-determined system of linear equation $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with $m < n$. Of course the solution is not unique. Nevertheless, let us solve it by minimizing the least square error

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \doteq \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2,$$

say using the simplest gradient descent algorithm:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha \nabla f\left(\boldsymbol{x}^k\right).$$

Show that if we initialize $\boldsymbol{x}_0$ as the origin $\mathbf{0}$, then when the above gradient descent algorithm converges, it must converge to the solution $\boldsymbol{x}_\star$ of the minimal 2-norm. That is, it converges to the optimal solution of the following problem

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_2^2 \quad \text{subject to} \quad \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}.$$

This is a phenomenon widely exploited in the practice of learning deep neural networks. Although due to over-parameterized, parameters that minimize the cost function might not be unique, the choice of optimization algorithms with proper initialization (here gradient descent starting from the origin) introduces implicit bias for the optimization path and converges to a desirable solution.

Exercise 6:
You are also welcome to implement some algorithms in Part 1 for other tasks.

A Additional Materials on Convexity

Below I briefly quoted some materials from the slide of Prof. Steve Boyd for your convenience. Check the textbooks provided in the lecture slide (Page. 12 of Part 0) for more details if you want to know more.

A.1 Practical methods for establishing convexity of a function

(i). verify definition

- $f(\theta \boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leq \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y}), \;\; 0 \leq \theta \leq 1$

(ii). restricting to a line and verify definition

- $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if the function $g : \mathbb{R} \to \mathbb{R}$,

$$g(t) = f(\boldsymbol{x} + t\boldsymbol{v}), \quad \operatorname{dom} g = \{t \mid \boldsymbol{x} + t\boldsymbol{v} \in \operatorname{dom} f\}$$

  is convex (in t) for any $\boldsymbol{x} \in \operatorname{dom} f, \boldsymbol{v} \in \mathbb{R}^n$

(iii). for twice differentiable functions

- show $\nabla^2 f(x) \succeq 0$ (i.e., the Hessian matrix $\nabla^2 f(x)$ is positive semidefinite)

(iv). show that f is obtained from simple convex functions by operations that preserve convexity

- nonnegative weighted sum
- composition with affine function ($g(\boldsymbol{x}) = f(A\boldsymbol{x} + b)$)
- pointwise maximum ($f(\boldsymbol{x}) = \max\{f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})\}$) and supremum
- perspective ($g(\boldsymbol{x}, t) = tf(\boldsymbol{x}/t) \;\; t > 0$)
- ...