1 Prologue

In today's fast-paced world driven by data, the ability to extract valuable insights and make informed decisions is more crucial than ever. Optimization, the process of finding the best solution among a set of alternatives, lies at the heart of this endeavor. From predicting customer behavior to optimizing supply chains, from designing machine learning models to solving complex decision-making problems, optimization techniques play a pivotal role in harnessing the power of data for practical applications.

In this course, we will explore the fundamental principles, algorithms, and applications of optimization. Through a blend of theory, practical examples, and hands-on exercises, we will equip ourselves with the necessary tools and techniques to tackle real-world optimization challenges in data-driven decision-making. There are no formal prerequisites, but basic knowledge of mathematical optimization and convex analysis will be assumed.

2 Introduction to optimization

We will get to learn the role of mathematical optimization. Here, a newcomer to the field may ask what mathematical optimization is. A mathematical optimization problem has the following canonical form.

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & x \in \mathcal{X} \end{aligned}$$

where

- x is referred to as the decision vector, the vector of decision variables, or simply the decision variables,
- $f(x)$ is the objective function that we want to optimize,
- $\mathcal{X}$ is the domain from which we may take values of the decision variables,
- $\min_x$ indicates that the goal of the optimization problem is to find and assigne values to the decision variables x *minimizing* the associated objective function value.

When we have $\max_x$ instead of $\min_x$, the goal is to *maximize* the objective function.

2.1 Example 1: Portfolio optimization

Given d financial assets (stocks, bonds, etc), we want to allocate x_i fraction of our budget to asset $i \in [d]$. Hence, we impose condition

$$1^\top x = 1.$$

Here, $x_i < 0$ indicates a *short position*, which means borrowing shares, selling now, and returning the shares later, while $x_i \geq 0$ indicates a *long position*, buying shares now. Then $\|x\|_1 = \sum_{i=1}^d |x_i|$ means *leverage*.

Let p_i be the initial price of asset i, and p_i' be its price at the end of one period. Then the return of asset i can be defined as

$$r_i := (p_i' - p_i)/p_i.$$

Moreover, the return of my portfolio can be measured by $r^\top x$. Here, r is a random variable with mean μ and covariance Σ. Then it follows that

$$\begin{aligned}\mathbb{E}\left[r^\top x\right] &= \mu^\top x \\ \mathrm{Var}\left[r^\top x\right] &= x^\top \Sigma x\end{aligned}$$

In words, the expected return, which is the expectation of the return $r^\top x$, is given by $\mu^\top x$. Moreover, the *risk* of my portfolio, which is often defined by the variance of the return $r^\top x$, is given by $x^\top \Sigma x$. By definition, the covariance matrix is positive semidefinite.

We want to find a portfolio that maximizes the expected return while guaranteeing a low risk. Then we consider

$$\begin{array}{ll}\text{maximize} & \mu^\top x - \gamma x^\top \Sigma x \\ \text{subject to} & 1^\top x = 1, \\ & x \in C'\end{array}$$

where $\gamma > 0$ is the risk aversion parameter. When $C' = \mathbb{R}^d_+$, we take long positions only. When $C' = \{x \in \mathbb{R}^d : \|x\|_1 \le B\}$, then we allow short positions but there is a leverage limit.

2.2 Example 2: Facility location

Given the locations of n households $x_1, \ldots, x_n \in \mathbb{R}^d$, we wish to build a hospital covering the households. A desired location would minimize the longest distance to a household. The problem can be formulated as

$$\min_x \quad \max_{i=1,\ldots,n} \|x - x_i\|.$$

2.3 Example 3: Supervised learning

A learning agent has access to a set of data $(x_1, y_1), \ldots, (x_n, y_n)$ where x_i denotes the *feature* and y_i is the corresponding *label*. For example, in image classification, x_i encodes the pixel values of the ith image file, and y_i indicates whether the image is a picture of a cat or a dog. Based on the data set, the learning agent's goal is to find a classifier or a regression function for the supervised learning task. Suppose that we have a hypothesis class $\mathcal{H} = \{h_1, h_2, \ldots\}$ where each h_j maps features to labels. A typical framework for modeling the supervised learning task is formulated as the following optimization problem.

$$\widehat{h} \in \operatorname*{argmin}_{h \in \mathcal{H}} \quad \frac{1}{n} \sum_{i=1}^{n} \ell\left(h(x_i), y_i\right)$$

where $\ell(y, y')$ is a loss function, e.g., $\ell(y, y') = (y - y')^2$, that accounts for the performance of classifier h on the data set. The optimization model provides an abstract and general framework for supervised learning. In the next section, we will see concrete examples of optimization models for supervised learning.

3 Optimization for machine learning

In the previous section, we have seen several examples of optimization models. Among them, supervised learning is a cenral topic in modern data science and machine learning. This section expands our discussion on optimization models for supervised learning and other related topics in machine learning.

3.1 Linear regression

Let us start by discussing linear regression. Linear regression is an example of supervised learning. Basically, given the predictor variable vector $x \in \mathbb{R}^d$, our hypothesis is that the response variable $y \in \mathbb{R}$ satisfies

$$\mathbb{E}[y \mid x] = w^{*\top} x \tag{1.1}$$

for some $w^* \in \mathbb{R}^d$. Given a set of data $(x_1, y_1), \ldots, (x_n, y_n)$, the goal is to infer the vector of coefficients w governing the relationship between the predictor variables and the response variable. We may propose a candidate vector w, which incurs the following.

$$\begin{aligned} \text{error} &= |y_i - w^\top x_i|, \quad i = 1, \ldots, n \\ \text{squared error} &= (y_i - w^\top x_i)^2, \quad i = 1, \ldots, n \\ \text{mean squared error} &= \frac{1}{n} \sum_{i=1}^{n} (y_i - w^\top x_i)^2 \end{aligned}$$

Here, the squared error comes from the squared loss function $\ell(y, y') = (y - y')^2$. Then the mean squared error measures how well the candidate vector w represents the data set. To deduce a coefficient vector that performs the best on the data set in terms of the mean squared error, we may solve

$$\min_{w \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} (y_i - w^\top x_i)^2. \tag{1.2}$$

In practice, the simple model (1.2) often results in several issuses such as *overfitting* and fails to detect *colinear* variables. To remedy these issues, a common practice is to introduce a regularization term in the objective. One popular way is to consider

$$\min_{w \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} (y_i - w^\top x_i)^2 + \lambda \|w\|_1 \tag{1.3}$$

for some $\lambda > 0$. Here, the regularization term $\lambda \|w\|_1$ induces *sparsity*, because the objective would rule out a vector w with large $\|w\|_1$. In practice, a sparse vector often achieves better results.

We have just described two optimization models (1.2) and (1.3) for linear regression, solving which returns a vector w to infer the true coefficient vector w^*. Then how can we solve the optimization models? In fact, the mean squared loss and the mean squared loss with the ℓ_1 regularization term are both *convex* functions of w. Therefore, (1.2) and (1.3) are both *convex optimization* problems. Convex optimization is fairly well understood, and we now have a wide range of algorithms and methods for convex optimization. For example, we may use FISTA, and more generally accelerated proximal gradient, for (1.3).

3.2 Neural networks and nonconvex optimization

Although the linear regression framework lays a stepping stone for analyzing the relationship between the predictor variables and the response variable, the linear model is often too restrictive in practical applications. In modern data science, *neural networks* are commonly used to solve a supervised learning task. For simplicity, let us focus on a neural network with a single hidden layer (Figure 1.1). Basically, given the predictor variable vector $x \in \mathbb{R}^d$, our hypothesis is that the

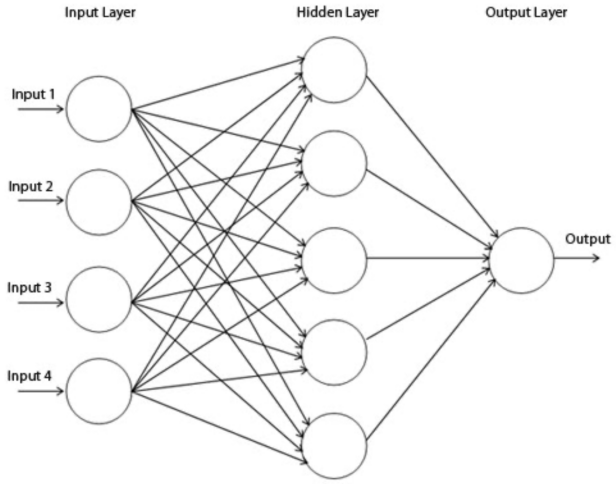

Figure 1.1: Single hidden layer neural network

response variable $y \in \mathbb{R}$ satisfies

$$\mathbb{E}\left[y \mid x\right] = w_2^\top \sigma(W_1^\top x) \tag{1.4}$$

where

- $W_1^\top x$ is the output of the input layer,
- σ is an activation function,
- w_2 is the weight vector that the hidden layer applies.

As linear regression, we may consider the mean squared error, which gives rise to

$$\min_{W_1, w_2} \quad \frac{1}{n} \sum_{i=1}^{n} \left(y_i - w_2^\top \sigma(W_1^\top x_i) \right)^2 . \tag{1.5}$$

Here, the objective function of (1.5) may not be convex depending on the structure of the activation function σ. ReLU and the sigmoid function are common choices for σ, and it is known that these activation functions lead to nonconvex objective functions. Hence, (1.5) is an instance of *nonconvex optimization*.

Nonconvex optimization is generally hard to solve. However, in practice, we often use *stochastic gradient descent (SGD)* and its variants to solve nonconvex optimization problems arising from training neural networks. SGD and its variants are known to work well in practice, and they are widely used in training neural networks. In this course, we will learn the theory behind SGD and its variants.

3.3 Minimax optimization

The third topic we discuss is *minimax optimization* which is relavant to many cutting-edge technologies in data science such as *Sharness-Aware Minimization* and *Generative Adversarial Network (GAN)*. Suppose that the response variable $y \in \mathbb{R}$ given the predictor variable vector $x \in \mathbb{R}^d$ satisfies

$$\mathbb{E}[y \mid x] = h(w, x)$$

where $h(w, \cdot)$ is a function parameterized by w, e.g., $h(w, x) = w^\top x$ for linear regression and $h((W_1, w_2), x) = w_2^\top \sigma(W_1^\top x)$ for the single hidden layer neural network. As before, one may consider the mean squared loss

$$\min_{w} \quad \frac{1}{n} \sum_{i=1}^{n} (y_i - h(w, x_i))^2 . \tag{1.6}$$

However, it is often observed that this optimization approach results in suboptimal performance at test time.

Inspired by this challenge, one would be interested in finding a parameter vector w whose entire neighborhoods have uniformly low training loss value. To achieve this goal, one may consider the following robust optimization framework.

$$\min_{w} \quad \max_{\|\epsilon\|_2 \le \rho} \quad \frac{1}{n} \sum_{i=1}^{n} (y_i - h(w + \epsilon, x))^2 . \tag{1.7}$$

One would also add a regularization term as follows.

$$\min_{w} \quad \max_{\|\epsilon\|_2 \le \rho} \quad \frac{1}{n} \sum_{i=1}^{n} (y_i - h(w + \epsilon, x))^2 + \lambda \|w\|_2^2. \tag{1.8}$$

The optimization framework (1.8) is referred to as Sharpness-Aware Minimization (SAM) introduced by Google Research in 2020 [FKMN21]. SAM is known to help prevent the model from overfitting to the training data and improves its generalization performance.

As another application of minimax optimization, we briefly mention Generative Adversarial Networks (GANs) [GPAM+14]. GANs are a powerful framework for training generative models through an adversarial process. In GANs, two neural networks, the generator and the discriminator, engage in a game-theoretic competition, akin to a minimax game. The generator aims to produce

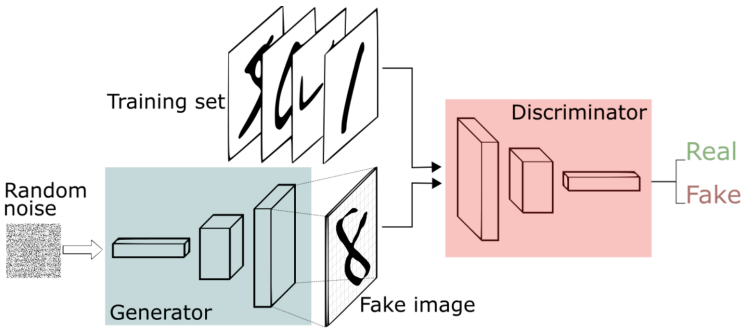

Figure 1.2: Simple description of Generative Adversarial Networks

realistic data samples that resemble those from a training dataset, while the discriminator aims to differentiate between real and fake samples. This adversarial dynamic drives both networks to improve continuously: the generator seeks to generate samples that are increasingly difficult for the discriminator to distinguish as fake, while the discriminator strives to become better at distinguishing real from fake samples. Through this adversarial process, GANs learn to generate high-quality, realistic data samples, with the generator gradually mastering the distribution of the real data. This minimax optimization framework underpinning GANs has revolutionized generative modeling, enabling remarkable advancements in generating realistic synthetic data across various domains.

3.4 Black-box optimization

Black-box optimization refers to a class of optimization problems where the objective function is treated as a black box, meaning that it is not explicitly defined or known. In other words, the function's analytical form or mathematical expression is unknown, and only its input-output behavior can be observed or evaluated. In particular, the objective function may be highly nonconvex, and

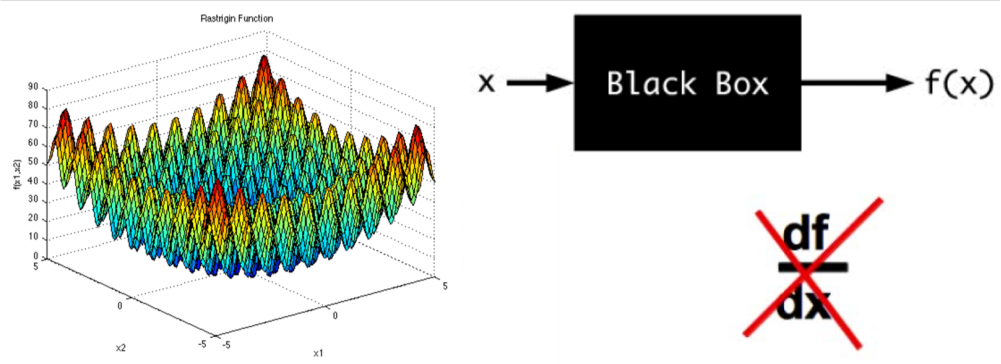

Figure 1.3: Simple description of Generative Adversarial Networks

we do not have access to the gradient information about the function.

In black-box optimization, the goal is to find the optimal input parameters that minimize or maximize the objective function without relying on its internal structure. This makes black-box optimization particularly useful in scenarios where the objective function is complex, expensive to evaluate, or involves noisy measurements.

Black-box optimization finds applications in a wide range of fields, including engineering design, machine learning, hyperparameter tuning, finance, and experimental design, where the underlying processes are complex or poorly understood, and traditional optimization approaches may not be suitable.

References

[FKMN21] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. 3.3

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In

Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. 3.3