

Lecture 1: The Basics of Optimization

*Lecturer: Swaprava Nath**Scribes: Aditya Singh, Dion Reji, Shreyas Katdare, Brian Mackwan*

Disclaimer: *These notes aggregate content from several texts and have not been subjected to the usual scrutiny deserved by formal publications. If you find errors, please bring to the notice of the Instructor.*

In this lecture, we discuss what essentially optimization is, where we use optimization, and look at a very basic optimization technique called linear programming.

Broadly, optimization problems can be classified into two classes:

Class	Variable	Solution space	Solution complexity
Continuous optimization	continuous	infinite	polynomial in the size of the problem
Discrete optimization	discrete	finite	exponential in the size of the problem

The knapsack problem is a classic discrete optimization problem, where we are given a bunch of objects with specific weights and a knapsack (backpack, for instance), which can carry at most some amount of weight. We want to fill our knapsack up to the highest permissible weight. There is nothing like partial association of an item with the knapsack: An item is either in the knapsack or not.

A polynomial solution to the most general knapsack problem is not known. We only have the brute force way of trying all possible combinations and finding the optimal one from that. In continuous optimization problems, even though the solution space is infinite, it is not very difficult to find the optimal solution. We will be talking about continuous optimization problems.

There is a significant trend in AI of formulating problems in terms of optimization problems.

1.1 What is optimization?

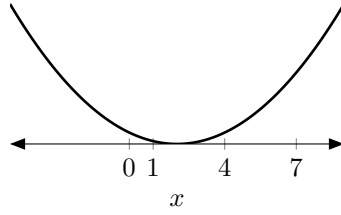
There are two components of an optimization problem: The objective function which we want to minimize (or maximize, which is just the negative of the minimization problem), and the constraint set. Let $f(x)$ be the objective function, and \mathcal{C} be the constraint set. Then, the optimization problem is written as

$$\min_{x \in \mathcal{C}} f(x).$$

Example. Minimize $(x - 2)^2$ with the constraint that $x \in [0, 1] \cup [4, 7]$.

Here, the objective function is $f(x) = (x - 2)^2$ and the constraint set is $\mathcal{C} = [0, 1] \cup [4, 7]$. This is a one-variable function, and we can easily see from the plot in figure 1.1 that $x^* = 1$ is the optimal value of x .

Example (Geometry). Suppose we have a map of a country with cities represented by the points $\mathbf{y}_1, \dots, \mathbf{y}_n$ in two-dimensions. We want to set up a supply chain and deliver to all these cities. We want to build a warehouse such that the sum of the distances from it to all the cities is minimized (assuming we can transport the Euclidean way).

Figure 1.1: A plot of $f(x) = (x - 2)^2$.

Let the warehouse be located at \mathbf{x} . Then, the optimization problem is

$$\min_{\mathbf{x} \in \mathcal{C}} \sum_{i=1}^m \|\mathbf{x} - \mathbf{y}_i\|_2,$$

where \mathcal{C} is the set of all points in the country, and $\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2}$ represents the L^2 norm of $\mathbf{x} = (x_1, x_2)$.

Example (Computer vision). Image de-blurring is a common problem in computer vision. We want to de-blur a blurred image according to some policy.

We consider grayscale images of size $m \times n$, where each pixel has an intensity value in $[0, 1]$: 0 meaning black, and 1 meaning white. Let $\mathbf{y} = [y_{i,j}]^{m \times n}$ be the blurred image that we are given. Let $\mathbf{x} = [x_{i,j}]^{m \times n}$ be the original image and \mathbf{k} be the blurring filter which was applied to get \mathbf{y} . Then, to get the original (de-blurred) image, we have to solve

$$\min_{\mathbf{x} \in [0,1]^{m \times n}} \left(\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |y_{i,j} - (\mathbf{k} * \mathbf{x})_{i,j}| + \lambda \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \underbrace{((x_{i,j} - x_{i,j+1})^2 + (x_{i+1,j} - x_{i,j})^2)}_{\text{to reduce sudden intensity changes}} \right).$$

λ and \mathbf{k} are the hyperparameters. λ penalizes a high difference in intensity of adjacent pixels.

Example (Machine learning). Suppose we have inputs (x_i, y_i) for $i \in [n]$. We are trying to fit a curve through these points. Suppose that we hypothesize the curve to be a polynomial $h_{\boldsymbol{\theta}}(x) = w_0 + w_1x + w_2x^2$, where $\boldsymbol{\theta} = (w_0, w_1, w_2)$ is a vector of the parameters. For each point x_i , we want the hypothesized point $h_{\boldsymbol{\theta}}(x_i)$ to be close to y_i . Let $\ell(x, y) = (x - y)^2$ be a (very simple) loss function. Our objective is to minimize the sum of the loss over all the observed points.

The optimization problem is

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^3} \sum_{i=1}^n \ell(h_{\boldsymbol{\theta}}(x_i), y_i).$$

1.1.1 Linear Programming: An optimization type

A linear program is a simple optimization type where the objective function and constraints are given by linear relationships.

Example (Political Winning). Investing money to win the election

Consider a political scenario where a political party P needs to invest money to win elections. There are 3 different demographic classes, Class 1, Class 2, and Class 3, and four issues to be addressed: A, B, C, and D. There is a specific pattern in which people from different Classes respond to different issues. The following

	Classes		
Issues	Class ₁	Class ₂	Class ₃
$x_1 \rightarrow A$	-2	5	3
$x_2 \rightarrow B$	8	2	-5
$x_3 \rightarrow C$	0	0	10
$x_4 \rightarrow D$	10	0	2
Population	100000	200000	50000
Majority	50000	100000	25000

table contains the number of votes gained/lost by the political party per unit money spent, on the respective Class and issue:

The population of different classes along with the majority in each Class required by the party to win the elections are also present in the table. The aim of the political party is to minimize the total amount of money it needs to invest, yet get the required majority across each Class. Let x_1, x_2, x_3, x_4 be the amount of money the party invests in issues A, B, C, and D respectively. Hence, this problem becomes the following optimization problem:

We want to minimize $x_1 + x_2 + x_3 + x_4$ and hence, the money spent on the election.

$$-2x_1 + 8x_2 + 0x_3 + 10x_4 \geq 50000 \quad (1.1)$$

$$5x_1 + 2x_2 + 0x_3 + 0x_4 \geq 100000 \quad (1.2)$$

$$3x_1 + 5x_2 + 10x_3 + 2x_4 \geq 25000 \quad (1.3)$$

$$(1.4)$$

Here, $x_1, x_2, x_3, x_4 \geq 0$. Let us look at the optimal solution:

$$x_1^* = \frac{2050000}{111}, \quad x_2^* = \frac{425000}{111}, \quad x_3^* = 0, \quad x_4^* = \frac{625000}{111}.$$

The optimal value of $x_1 + x_2 + x_3 + x_4 = x_1^* + x_2^* + x_3^* + x_4^* = 3100000/111$.

After multiplying and adding the equations as $(1.1) \cdot 25/222 + (1.2) \cdot 46/222 + (1.3) \cdot 14/222$, we get

$$x_1 + x_2 + \frac{140}{222}x_3 + x_4 \geq \frac{3100000}{111}$$

since $x_1 + x_2 + x_3 + x_4 \geq x_1 + x_2 + 140x_3/222 + x_4 \geq 3100000/111$. This proves that our given solution is truly the optimal solution.

1.1.2 Standard Form of Linear Program

Let \mathbf{x} be the vector containing variables to optimize and \mathbf{c} be the vector of constants:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}.$$

Then, we can write the standard form of linear program as $\max \mathbf{c}^\top \mathbf{x}$ subject to the constraints

$$\mathbf{A}_{m \times n} \mathbf{x}_{n \times 1} \leq \mathbf{b}_{m \times 1} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix},$$

where $\mathbf{x} \geq \mathbf{0}$. \geq represents element-wise greater than or equal to:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \iff \forall i, x_i \geq 0.$$

The aforementioned problem is commonly referred to as the primal problem, and it is accompanied by a corresponding dual problem.

Primal Problem (P1)

$$\begin{aligned} &\text{maximize} && \mathbf{c}^\top \mathbf{x} \\ &\text{subject to} && \mathbf{Ax} \leq \mathbf{b} \\ &&& \mathbf{x} \geq \mathbf{0} \end{aligned}$$

Dual Problem (P2)

$$\begin{aligned} &\text{minimize} && \mathbf{b}^\top \mathbf{y} \\ &\text{subject to} && \mathbf{A}^\top \mathbf{y} \geq \mathbf{c} \\ &&& \mathbf{y} \geq \mathbf{0} \end{aligned}$$

1.1.3 Weak Duality Principle

Theorem 1.1 (Weak Duality Principle) *Let \mathbf{x} and \mathbf{y} represent feasible solutions, i.e., solutions that satisfy all the constraints, for the primal and dual problems, respectively. Then,*

$$\mathbf{b}^\top \mathbf{y} \geq \mathbf{c}^\top \mathbf{x}.$$

Proof.

$$\mathbf{Ax} \leq \mathbf{b} \tag{1.5}$$

$$\mathbf{x}^\top \mathbf{A}^\top \leq \mathbf{b}^\top \tag{1.6}$$

As $\mathbf{y} \geq \mathbf{0}$, multiplying it on both sides will not change the inequality:

$$\mathbf{x}^\top \mathbf{A}^\top \mathbf{y} \leq \mathbf{b}^\top \mathbf{y} \tag{1.7}$$

Since $\mathbf{A}^\top \mathbf{y} \geq \mathbf{c}$,

$$\mathbf{b}^\top \mathbf{y} \geq \mathbf{x}^\top \mathbf{c}, \tag{1.8}$$

which can also be written as $\mathbf{b}^\top \mathbf{y} \geq \mathbf{c}^\top \mathbf{x}$. Thus, weak duality principle provides a relation between the solutions of primal and dual problems. ■

1.1.4 Strong Duality Theorem

If a linear programming problem has an optimal solution, so does its dual. If \mathbf{x}^* and \mathbf{y}^* are the optimal solutions of the primal and dual problems respectively, then

$$\mathbf{b}^\top \mathbf{y}^* = \mathbf{c}^\top \mathbf{x}^*.$$