1 Optimization Bootcamp

I write this as my flight approaches Charles de Gaulle airport in Paris, and I am overwhelmed by signs of optimization. The aircraft itself is a pinnacle of modern engineering optimization. Flying at nearly the speed of sound, the shape of the wings has been carefully refined for efficient flight at cruise and to provide enough lift for a heavy, fully fueled take off. The composite materials used to construct the wings and body were designed for strength and durability, and to provide higher cabin pressure and moisture to reduce hypoxia and jetlag for its passengers. Further, the aircraft has been optimized not only for flight, but for global manufacturability, as it is assembled from supply chains spanning several continents. Out the window, I see farmlands yield to city, a quiet scene of distributed optimization; fields and water management transition into planned streets and urban arterials, ensuring the city is fed. The surrounding landscape has been shaped by the scars of war, a terrible reminder of the birth of modern optimization. As we near the city, I see hills that did not exist in Roman times, signs of Napoleon's great earth-moving efforts and the birth of optimal transport[1]. The airport is likewise a city of its own: supply chains, inventories, ground crews, flight controllers, food service workers, and custodians working in shifts, all in a vast system of local optimization. And I write this on my laptop[2], which is a compact masterpiece of constrained optimization.

Our world is built on optimization. How did we get here, and what are the tools of modern, applied optimization? This chapter provides an introductory overview of key concepts in optimization, as well as several motivating applications. Although this chapter is not meant to be entirely standalone, it serves as the bulk of an initial *bootcamp* course, augmented with deeper dives into each topic in later chapters. Many of the sections here will be explored in much more detail in later chapters, and so this is mostly meant to provide a rapid overview and categorization of the major topics in optimization, important algorithms, and how the various ingredients fit together. Thus, this is also meant to be a preview of later chapters.

We begin with a categorization of different optimization problems, the so-called *anatomy* of an optimization problem. We introduce the objective function, to be maximized or minimized, and constraint equations that limit the set of feasible solutions.

[1] My favorite café in the city is on Rue Monge, named after Gaspard Monge, who developed optimal transport to efficiently reshape Paris at Napoleon's command.

[2] This book was an exercise in constrained optimization. My objective: to write the most interesting and informative book possible for scientists and engineers within the constraints of one year and 500 pages.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

Next we introduce two of the main concepts in optimization: convexity and the gradient. Convexity provides a fundamental dichotomy in optimization, where convex problems are essentially solved, at least numerically, and where local optimal solutions are also globally optimal. Convexity is used throughout the book, although this is the only chapter that formally discusses the conditions for convexity and its implications for global optimization. Chapters 3 and 4 on linear programming and quadratic programming (e.g., least-squares regression) cover two broad classes of convex optimization problems. Nonconvex problems are often more challenging, and do not generically have solutions that scale computationally. However, there has been a major shift in applied optimization, where it is increasingly possible to directly solve nonconvex problems, rather than relaxing them to nearby convex problems. Although there are fewer guarantees, solving the nonconvex problem directly is often highly effective in practice. Chapter 5 explores modern approaches to nonconvex optimization problems in more detail.

The gradient is another fundamental concept in optimization, as it provides critical information about the geometry of the optimization landscape, and which direction (locally) improves the solution. This topic is explored in much greater depth in Chapter 2, including algorithms that use gradient descent, Newton's method, and modern extensions for high-dimensional systems, such as stochastic gradient descent in machine learning.

Throughout the chapter, we examine examples of unconstrained and constrained optimization, to remind the reader of basic properties and approaches. These will be simple low-dimensional optimization problems, where it is possible to solve for the optimum solution using basic calculus. We also briefly introduce the notion of Lagrange multipliers, which are an intuitive and physical way of handling constraint equations. Lagrange multipliers are closely related to the notion of a dual optimization problem, and both concepts are developed further in Chapter 6.

Modern mathematical optimization opens up a world of applications in science, engineering, and industry. We conclude with a brief look at three major application areas of modern optimization: inverse problems,, control theory, and machine learning. For example, nearly all of machine learning is built around optimization, which is used to train models to fit data. Similarly, control theory is essentially a field of applied optimization, where the problem is constrained by a differential equation that captures the underlying dynamic behavior of a system. Inverse problems form another major class of applied optimization problems, where the goal is to infer unknown quantities or parameters of a model to best match observed data. Each of these topics will be their own chapter in the later half of this book, with an additional chapter on Bayesian optimization to complement the topics of inverse problems and machine learning.

On your first read, this chapter may be skimmed, and I encourage you to peek into later topics in the book. Some of the technical results around convexity can also be skipped, especially the proofs. However, the big-picture idea that convexity leads to powerful guarantees and efficient algorithms is an important message of this chapter. But remember that convexity isn't everything, that most of the real world has horns and fangs, and that we must be ready to face nonconvexity in its many forms. The rest is meant to pique your interest and provide you with a road map for further studies. Bon voyage.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

1.1 Anatomy of an Optimization Problem

In its simplest form, an optimization problem involves minimizing or maximizing an objective function $f(\mathbf{x})$ with respect to a set of decision variables $\mathbf{x} \in \mathbb{R}^n$:

$$\min_{\mathbf{x}} \quad f(\mathbf{x}). \tag{1.1}$$

This is an *unconstrained* optimization problem, as it is assumed that the decision variable may take any value. For scalar functions of a scalar variable, we have experience solving these optimization problems by finding points where the derivative of f with respect to x is equal to zero that also have a positive second derivative. See Panel 1.1–Panel 1.3.

However, many real-world optimization problems do have constraints on $\mathbf{x}$, which are written either as inequality or equality constraint equations. In Panel 1.1, we implicitly assumed that the angle α should be constrained to be between 0 and $\pi/2$. Panel 1.1–Panel 1.3 also admit natural constraints, such as restricted lake crossing zone, or positivity constraints on the coefficients of the model. In general, a *constrained optimization* optimization problem over a variable $\mathbf{x}$ may be written in the following *standard* form:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}), \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m, \\ & h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, p, \end{aligned} \tag{1.2}$$

Panel 1.1: Maximizing Projectile Distance

First, let's solve for the best angle α to launch a projectile with initial speed v to maximize its range. The distance traveled is $D(\alpha) = \frac{v^2}{g}\sin(2\alpha)$.

To derive the total distance, we decompose v into the vertical velocity $v_y = v\sin(\alpha)$, and the horizontal velocity $v_x = v\cos(\alpha)$. The acceleration due to gravity is $-g$, so the vertical position is $y(t) = v\sin(\alpha)t - \frac{g}{2}t^2$. The ball lands on the ground again after time T, so $y(T) = 0$ so $T = \frac{2v}{g}\sin(\alpha)$. In this time, the projectile travels $D(\alpha) = Tv\cos(\alpha) = \frac{2v^2}{g}\sin(\alpha)\cos(\alpha) = \frac{v^2}{g}\sin(2\alpha)$ in the horizontal direction.

We now optimize this with respect to α. At a maximizing α, changing α has no affect on the distance D, so the derivative $\mathrm{d}D(\alpha)/\mathrm{d}\alpha = \frac{2v^2}{g}\cos(2\alpha)$ must be equal to zero. Thus, $2\alpha = \pi/2$, which implies an optimal angle, α^*, equal to $\pi/4$.

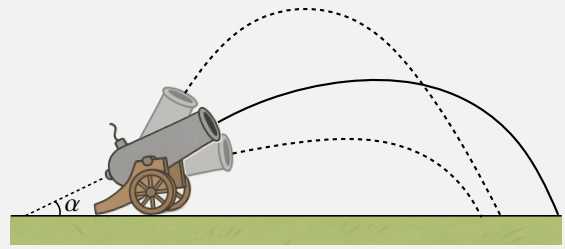


Figure 1.1: Schematic of canon trajectory for various launch angles α.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

Panel 1.2: Optimal Lake Crossing

A classic optimization problem is to find the fastest path between points A and B, where some distance is traveled on land, and some is traveled over a lake. If the distance was all traveled on level ground, we know that the shortest path is a straight line. However, when the land speed is v_1 and the swimming speed is $v_2 < v_1$, the solution becomes more interesting. Specifically, what is the optimal point x to cross the lake?

This problem may be approached several ways, which we will explore in the homework. This geometric problem is a simple, discrete cousin of the famous Brachistochrone problem, which is discussed in much greater detail in Section 6.6.

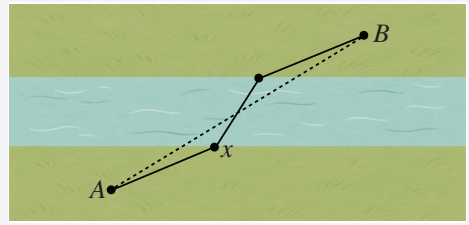


Figure 1.2: Schematic of the lake crossing problem.

Panel 1.3: Data Fitting

A more relevant example is fitting a model to describe data. In this case, we have a collection of data points $\{(x_j, y_j)\}_{j=1}^{n}$, and we optimize the parameters of either a linear or an exponential model. We explore data fitting as an optimization problem in Chapter 4 on least-squares regression and Chapter 10 on machine learning. The following code shows how simple it is to solve for a linear fit in Python.

```
import numpy as np
from scipy.optimize import minimize
x = np.linspace(0, 10, 20); y = 3*x+2 + np.random.randn(20)
def f(p):
    return np.sum((y - (p[0]*x + p[1]))**2) # least squares
res = minimize(f,[0,0]) # initial guess slope=0, intercept=0
a_opt, b_opt = res.x
```

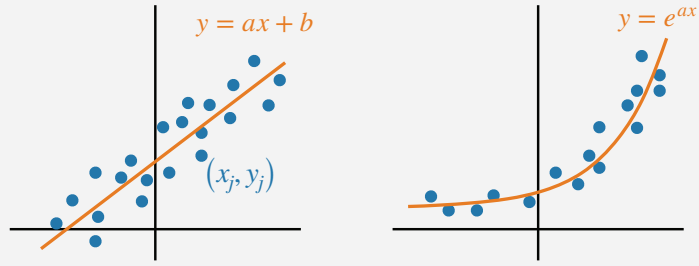


Figure 1.3: Examples of two data-fitting regression problems.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

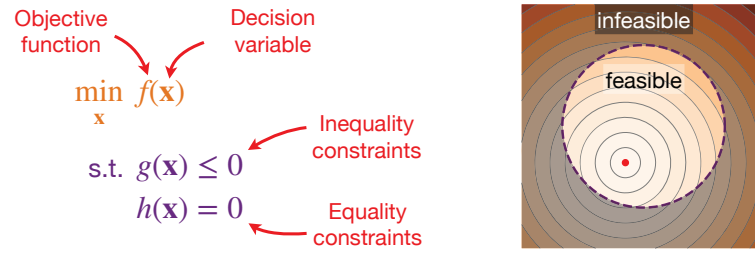


Figure 1.4 The anatomy of an optimization problem. The two main ingredients are the objective function f and the constraint equations g and h. We optimize over the decision variable $\mathbf{x}$. In the case on the right, the constraints are $g(x_1, x_2) = x_1^2 + x_2^2 \leq 1$ and the objective function is $f(x_1, x_2) = (x_1 + 0.5)^2 + (x_2 + 0.5)^2$. We visualize the objective function by logarithmically spaced contours, with the minimum shown as a red dot.

where $f(\mathbf{x})$ is the objective function to be minimized, and $g_i(\mathbf{x})$ and $h_j(\mathbf{x})$ are inequality and equality constraint equations that restrict the set of feasible $\mathbf{x}$ to search over. Typically the objective function f is a scalar function[3] over a vector input $\mathbf{x}$. Figure 1.4 shows this schematically, color coded with the colors used throughout this chapter.

We often write the constraint equations in vector form as:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}), \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ & \mathbf{h}(\mathbf{x}) = \mathbf{0}. \end{aligned} \tag{1.3}$$

The inequality and equality constraints determine a feasible set $\mathcal{S}$ of admissible values of $\mathbf{x}$ over which to optimize the objective function f. We use the symbol $\leq$ to indicate that each row of the system of equations must be an inequality; some texts use the symbol $\preccurlyeq$ for this. We also shorten *subject to* to *s.t.*. Several example constraints are considered in Panel 1.4.

When the objective function f and the feasible set $\mathcal{S}$ are both convex, then Eq. (1.2) is a *convex* optimization problem. Convex optimization problems come with strong theoretical guarantees and efficient computational algorithms, as we will discuss in Section 1.2. Linear programming and quadratic programming (e.g., least-squares regression) are two of the most important classes of convex optimization problems. The most important result is that a local minimum of a convex optimization problem is also a global minimum. If the problem is strictly convex, this minimum is unique. Convex sets and functions are discussed later in this chapter, with several examples; see Fig. 1.8 for examples of convex sets and functions.

Maximization is a trivial extension with similar theoretical guarantees when f is concave and the feasible set is convex. Such optimization problems are still considered *convex*, as maximizing a concave function f is equivalent to minimizing the convex function $-f$. For the special case of a linear program, where the linear objective function $f(\mathbf{x}) = \mathbf{c}^{\mathrm{T}}\mathbf{x}$ is both convex and concave, we may pose either minimization or maximization problems.

[3] In multi-objective optimization, the objective function has multiple components and is written as a vector function $\mathbf{f}$. In this case, there is a family of optimal solutions for different weightings of the components of $\mathbf{f}$, which are known as a Pareto front. This is common in engineering applications, such as aircraft design, where there are multiple competing design objectives, (e.g., size, weight, and power). See Panel 5.2.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

Panel 1.4: Simple Constraints

There are several intuitive examples of where constraints arise in optimization:

Packing problems: If we are packing a delivery truck, it may have a volume and weight capacity, given by $V_{\max}$ and $W_{\max}$, respectively. Then the total volume and weight of the cargo must be below these max values:

$$\sum_{j=1}^{n} v_j x_j \leq V_{\max} \qquad \text{and} \qquad \sum_{j=1}^{n} w_j x_j \leq W_{\max},$$

where x_j is the quantity of item j, v_j and w_j are the volume and weight of this item, respectively, and there are n distinct item types. These are *linear* inequality constraints of the form $\mathbf{c}^T\mathbf{x} \leq \mathbf{d}$.

Investment portfolio: Given a total budget B for an investment portfolio, with c_j the cost of asset j, and x_j the quantity purchased, we have an inequality constraint:

$$\sum_{j=1}^{n} c_j x_j \leq B.$$

Communication radius: If a drone must operate in a plane of radius R, where it is able to communicate to a base station, then we have a convex, quadratic constraint:

$$x^2 + y^2 \leq R^2.$$

Controllability ellipsoid: If we are controlling a linear dynamical system and have one unit of control input (e.g., one liter of gasoline), then the state is constrained by an ellipsoid defined by the controllability Gramian (see Chapter 9 and [77]):

$$\mathbf{x}^T\mathbf{W}\mathbf{x} \leq 1.$$

Legal and regulatory constraints: Many businesses, for example in finance, operate within the bounds of legal and regulatory constraints. And as is the nature of constraints in optimization problems, we find that optimal solutions edge right up against the boundary of these constraints.

Standard Form Optimization Problems

One of my goals is for you to learn how to recognize an optimization problem, identify the basic problem template, and choose which solution method to use. There are several standard problems in optimization that we will revisit throughout this book. Two of the most well-studied problems are those of linear programming and quadratic programming. These are the topics of Chapters 3 and 4, respectively.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

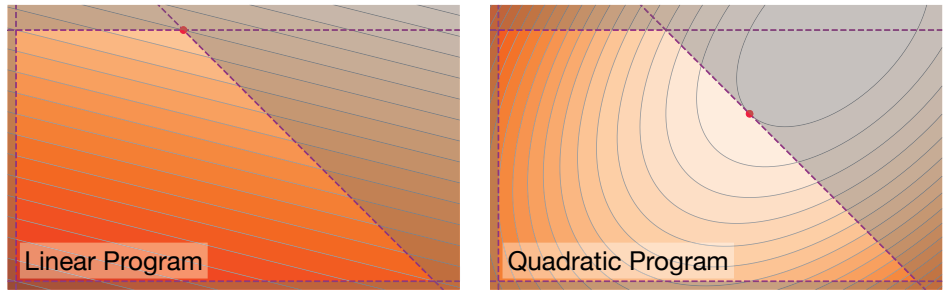


Figure 1.5 Two standard optimization problems are linear programming (left) and quadratic programming (right). These are covered in Chapters 3 and 4, respectively. Linear programs are always convex, and quadratic programs are convex if the matrix $\mathbf{Q}$ in the quadratic objective function is positive semi-definite. In linear programming, when the feasible region is a closed set, the optimal is typically at a vertex of the feasible polytope. In quadratic programming, the optimal point may be in the interior of the feasible region, or at one of the boundaries.

Linear Programming (LP)

Linear programming involves optimizing a linear objective function subject to linear equality and inequality constraints.

Let's construct a simple example. Imagine we are trying to maximize the profit associated with manufacturing two types of handbags: we produce x_1 units of a simple bag with profit $c_1 = \$100$ per unit, and x_2 units of a luxury with profit $c_2 = \$400$ per unit. The total production capacity of our factory means that the total number of units produced must be no greater than 50 and the number of units of item two must be no greater than 30. Finally, we cannot produce a negative number of units, so $\mathbf{x} \geq 0$.

This may be encoded into an optimization problem of the following form:

$$\begin{aligned} \max_{\mathbf{x}} \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x}, \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{1.4}$$

Here $\mathbf{c} \in \mathbb{R}^n$ is the cost vector, while $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$ define the constraints. For the example above, we have

$$\mathbf{c}^{\mathrm{T}} = \begin{bmatrix} 100 & 400 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 50 \\ 30 \end{bmatrix}.$$

The inequality constraints determine a convex feasible region $\mathcal{S}$, called a polytope, that generalizes the notion of a polygon to higher dimensions. The objective function $f(\mathbf{x}) = \mathbf{c}^{\mathrm{T}}\mathbf{x}$ is also convex, making this a convex optimization problem. The linear objective function and convex feasible region are shown in Fig. 1.5 (left).

Note that minimization and maximization problems are both convex, because the linear objective function is both convex and concave, a concept we will explore in a moment.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

Quadratic Programming (QP)

Quadratic programming optimizes a quadratic objective subject to linear constraints:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{c}^{\mathrm{T}}\mathbf{x}, \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{1.5}$$

Quadratic programs are widely used in finance and control theory, and they are the basis of least-squares regression in data analysis and machine learning. When the matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is a symmetric positive semi-definite matrix, then the quadratic program is *convex*. The quadratic objective function and convex feasible region are shown in Fig. 1.5 (right).

As a concrete example, consider the least-squares regression problem, where we solve for $\mathbf{x}$ to minimize the mean-squared error (MSE)[4] of the fit $\mathbf{A}\mathbf{x} \approx \mathbf{b}$, defined as:

$$\min_{\mathbf{x}} \tfrac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2. \tag{1.6}$$

We may expand this out as $f(\mathbf{x}) = \frac{1}{2}(\mathbf{A}\mathbf{x} - \mathbf{b})^{\mathrm{T}}(\mathbf{A}\mathbf{x} - \mathbf{b}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x} - \mathbf{b}^{\mathrm{T}}\mathbf{A}\mathbf{x} + \frac{1}{2}\mathbf{b}^{\mathrm{T}}\mathbf{b}$. Since the constant $\mathbf{b}^{\mathrm{T}}\mathbf{b}$ doesn't affect the optimization problem, we may remove it, and the objective function now has the form in Eq. (1.5), with $\mathbf{Q} = \mathbf{A}^{\mathrm{T}}\mathbf{A}$ and $\mathbf{c}^{\mathrm{T}} = -\mathbf{b}^{\mathrm{T}}\mathbf{A}$. The $\mathbf{Q}$ matrix in this case is symmetric and positive definite, so the problem is convex, as we will see soon. We may solve this problem with or without constraints, as explored in Chapter 4.

The data fitting problem in Panel 1.3 is a simple least squares regression problem with a quadratic objective function. In this case, the matrix $\mathbf{A}$ and vectors $\mathbf{x}$ and $\mathbf{b}$ are

$$\mathbf{A} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} a \\ b \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

The Objective Function FAQ

We often solve unconstrained optimization problems, in which case we are only concerned with the properties of the objective function f. In either the constrained or unconstrained cases, there are several important properties of f that determine which optimization algorithms are well suited to solve the problem.

Is f Convex or Nonconvex? This is the biggest dichotomy of optimization problems. Convex optimization problems have strong theoretical guarantees and powerful algorithms to find the globally optimal solution, even for large-scale problems. We carefully explore what makes an optimization problem convex in this chapter, and we will use this perspective throughout the remainder of the book.

[4] Here, the MSE is given by the square of the ℓ_2 norm, defined as $\|\mathbf{x}\|_2^2 = \sum_{j=1}^{n} x_j^2$. For $\mathbf{x} \in \mathbb{R}^2$ this becomes $\|\mathbf{x}\|_2^2 = x_1^2 + x_2^2$. We discuss norms in much greater detail later in this chapter.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

Examples: The linear least-squares regression problem from Chapter 4 is a good example of a convex optimization problem, where there is a single, globally optimal solution. It is even possible to analytically derive an expression for this optimum. Nonlinear least-squares regression, such as training a neural network in Chapter 10 is an example of a nonconvex optimization problem. In this case, the objective function has many local minima and saddle points, making it challenging to find a global optimum.

Can We Compute the Gradient $\nabla \mathbf{f}$? Having access to the gradient ∇f allows us to use gradient descent and its variants to find candidate local minima where $\nabla f = \mathbf{0}$. Higher-order derivatives may be used for Newton's method, which has faster convergence rates. Both are discussed in Chapter 2. Even when f is nonsmooth, it may be possible to compute the *proximal gradient*, which is discussed in Section 5.1.

Examples: Sometimes we have an analytic expression for the gradient of f, as in least-squares regression (see Chapter 4). In other cases, we approximate the gradient numerically, for example using finite difference derivatives. When training a neural network, we use stochastic gradient descent and estimate the gradient via backpropagation.

In some cases, gradients are difficult to estimate or do not even exist. For example, some objective functions are nonsmooth and do not admit gradient descent methods. In other cases, we may be optimizing some physical process where we do not have a model of f to differentiate. For example, when optimizing the shape of a compressor blade in an experiment or simulation, we may not have access to a differentiable model of the process, and instead must resort to gradient-free iterative optimization. Gradient-free optimization techniques are typically much more expensive and inefficient; these approaches are explored further in Chapter 5.

In fact gradient-free optimization is so expensive that we often go to great lengths to come up with gradients, for example by developing smooth surrogate models f or developing adjoint simulations for complex shape optimization problems.

Is $\mathbf{f}$ Linear or Nonlinear? The structure of the function f plays a major role in the optimization landscape, and this is closely related to convexity above. We will consider several classes of functions f, including linear, quadratic, and more general nonlinear functions. Linear functions are convex, but there is not a clear connection between nonlinearity of f and nonconvexity. However, highly nonlinear functions, such as neural network training landscapes, are more likely to have many local minima and be nonconvex.

Examples: If f is linear, as in linear programming (see Chapter 3) then the objective function is flat, so it is both convex and concave. It is possible to compute the gradient of a linear objective function analytically, and the maximum or minimum of the function depends entirely on the geometry of the feasible set given by the constraint equations.

If f is quadratic, we may also compute the gradient of f analytically, as in Chapter 4. However, the convexity of f now depends on the Hessian matrix $\mathbf{H}$ of second partial derivatives. If $\mathbf{H}$ is positive definite, so that all its eigenvalues are positive, then f has

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

Panel 1.5: Minima, Maxima, and Saddle Points

An objective function $f(x)$ may have minima, maxima, and saddle points.

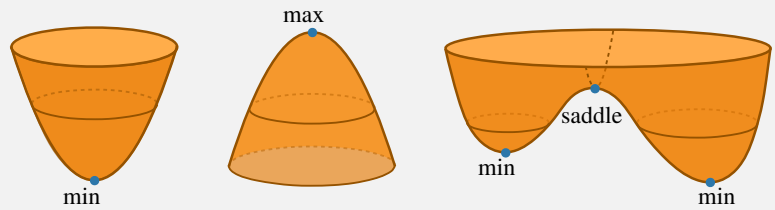


Figure 1.6: Examples of various convex, concave, and indeterminant functions.

positive curvature in every direction $\mathbf{x}$ and is convex. The linear least-squares problem discussed above is a quadratic objective function f with positive definite Hessian $\mathbf{H}$. If $\mathbf{H}$ is negative definite, then f is concave, and if it is indefinite, so that it has both negative and positive eigenvalues, then f has a saddle point and is neither convex nor concave.

For more general nonlinear f, polynomial and otherwise, the function may or may not be well behaved. For example, the function $f(x) = \exp(x)$ is convex with analytical derivative; however, the minimum of $\exp(x)$ occurs at $x \to -\infty$, so it is never attained. In contrast, the function $f(x) = \sin(x)$ has infinitely many local minima and maxima. However, both functions are relatively easy to work with and it is straightforward to compute derivatives and solve for extrema. In higher dimensions, however, enumerating and characterizing the extrema may become quite challenging or intractable.

Is f High Dimensional? Many modern problems involve very high-dimensional decision variables $\mathbf{x}$, so that gradients and Hessians of f are high-dimensional. In general, high-dimensional optimization problems are more computationally expensive and are prone to ill-conditioned solutions.

Examples: In deep learning, it is not uncommon to train a model with millions or billions of free parameters to be optimized over. This motivates a certain class of algorithm, such as stochastic gradient descent. Even for convex problems, when the dimension is large, the optimization problem may be ill-conditioned or slow to converge. For example, large-scale linear least-squares problems are often ill-conditioned and require regularization techniques to improve the quality of the optimization solution. Many modern optimization packages and algorithms, such as interior point methods (IPMs) are designed specifically to scale robustly to high-dimensional problems. These issues are discussed in Chapters 4 and 6. Further, the issues associated with nonconvex optimization tend to compound in higher dimensions, as the number of possible local minima to be checked may grow exponentially, or even combinatorially, in the dimension of $\mathbf{x}$.

Is f Known or Unknown? In many modern applications, there is not a mathematical model for f, and it must be computed by interacting with a complicated simulation or

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

experiment. When f is not easily representable as an analytical function, or a function that can be represented with a computational graph, then the solution techniques must be modified. For example, if f is a black box simulation or experiment, we may be forced to resort to gradient-free optimization techniques in Chapter 5. However, we are increasingly able to construct data-driven surrogate models for such f using machine learning, and these surrogate models can be used to accelerate the optimization process.

Examples: In many engineering optimization problems, our model of the real world is imperfect, or too expensive to easily model or simulate in a function f. For example, when designing a Formula One car, evaluating a new test design may involve fabricating a model and testing for hours in a wind tunnel. Thus, we may need to optimize over a physical system or experiment, for which there is no well behaved function f. Similarly, we may be optimizing a simulation where some components of the simulation are legacy compiled codes and we no longer have access to the source code.

Is f Expensive to Compute or Evaluate? This is closely related to the previous issue. When the function f is expensive to evaluate, we typically must be more conscientious of how our optimization algorithm scales with function evaluations. Iterative methods, such as gradient descent and Newton's method, have well-understood scaling properties, and the computational cost of f often guides which method to use. Of course, the expense of computing f may be related to the dimension of $\mathbf{x}$ and the nonlinearity of f, compounding with the issues above.

Examples: In many engineering applications, such as aerodynamic shape optimization of a wing, it is expensive to evaluate the objective function for a single $\mathbf{x}$. This may involve a computationally intensive, high-fidelity simulation, or actually fabricating and testing the wing in a wind tunnel. When function evaluation is expensive, we require algorithms that converge in very few iterations, which often motivate surrogate models.

Does f Encode Dynamics in Time? When the system being optimized involves time dynamics, this often changes the optimization algorithm and computational cost involved. In this case, the dynamics of f must be captured via differential equations, which are typically more involved, both conceptually and computationally. Alternatively, we may average out the time dynamics and optimize over mean quantities, although this may also be expensive, requiring enough time for a converged average.

Examples: Control theory (see Chapter 9) is a good example where we optimize a control law to modify the performance of a dynamical system, which inherently varies in time. Other optimization problems, such as aerodynamic shape optimization, may involve time averaging dynamic effects, such as turbulent fluctuations. Many optimization problems we face do not explicitly involve time dynamics, such as packing or resource allocation problems in Chapter 3 or regression problems in Chapter 4.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

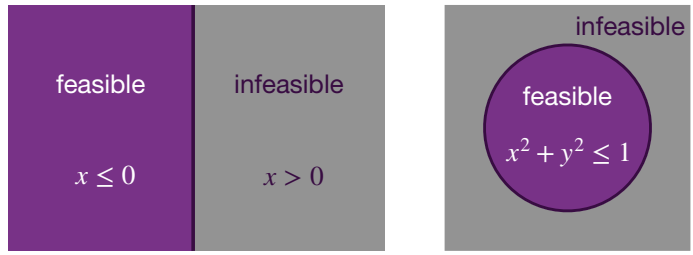


Figure 1.7 Examples of feasible and infeasible sets determined by inequalities. A single linear inequality defines a convex half-space (left); multiple linear inequalities result in intersecting half-spaces and define a polytope (see Fig. 1.11). Nonlinear inequalities define more complex feasible sets (right). In this case, the feasible region is still convex.

The Constraint Equations

Constraint equations arise naturally in optimization. Any time a decision variable involves a constrained resource, such as time, space, money, materials, labor, etc., these may be expressed as constraint equations. This is common in linear programming, which involves minimizing or maximizing an objective function under linearly constrained resources.

We will see many other types of constraints throughout this book. For example, we may want to solve a least-squares regression problem where we know that some variables must be non-negative, or within some radius (see Fig. 1.7). Control theory is another example, where the constraints are governed by the dynamics of the system.

Figure 1.7 shows two examples of feasible sets given by inequality constraints. The first example is a linear inequality constraint, and the second example is a nonlinear inequality constraint. Any linear inequality constraint, given by $g(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \leq 0$, determines a feasible half-plane in the n-dimensional space. The complementary inequality $\mathbf{c}^T\mathbf{x} > 0$ determines an infeasible half plane. The nonlinear constraint $g(\mathbf{x}) = x^2 + y^2 - 1$ determines a feasible circular disk of radius equal to 1, with an infeasible region exterior to this disk. In both cases, the feasible region is convex, as we will see defined formally soon.

The Feasibility Problem

In nearly every domain using optimization – engineering design with physical constraints, finance with regulatory constraints, or machine learning with interpretability constraints – maximizing or minimizing an objective is meaningless if we cannot locate any feasible solutions. Thus, verifying feasibility is a foundational step before optimizing a solution.

For high-dimensional problems with complicated constraint equations, even finding a feasible $\mathbf{x}$ might be a challenge. The *feasibility problem* is

$$\begin{aligned} \text{find} \quad & \mathbf{x}, \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ & \mathbf{h}(\mathbf{x}) = \mathbf{0}. \end{aligned} \tag{1.7}$$

Ensuring feasibility (i.e., satisfying all constraints) can be at least as challenging as optimizing an objective. One might have to solve large sets of equations or inequalities to find a point that meets all physical, geometric, or statistical constraints, before even

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

addressing optimality. We will see an example of a feasible set obtained via the intersection of several inequality constraints later, in Fig. 1.11, in the context of linear programming.

Feasibility is challenging because constraints may define complicated sets. High-dimensional constraints can intersect in to cause narrow or fragmented feasible sets. Sometimes it is unknown if a feasible region even exists. Many optimization problems are ill-posed, so small changes to the problem can dramatically change the feasible set. For example, in inverse problems, one must find model parameters consistent with observed data, enforcing model constraints (e.g., PDEs or boundary conditions). Feasible solutions are sensitive to these constraints. Similarly, in control theory, a controller must often satisfy closed-loop stability, performance bounds, or actuator limits. Ensuring a feasible controller even exists can be challenging, especially if the system is barely controllable.

There are several strategies for approaching the feasibility problem. For linear or convex constraints, there are specialized approaches, such as linear matrix inequality solvers, that can systematically search for feasible solutions. For more complex constraints, we may start with a simpler version of the constraints and continuously transition to the more challenging constraints, using homotopy or continuation to track the solutions through this transition. Other approaches involve penalty or barrier methods that transform constraints into large penalty terms in the objective, promoting feasible solutions. Projection and iterative methods, such as Douglas–Rachford methods [160], may also be used to project guesses onto a feasible set; these are often used in PDE-constrained problems.

Box Constraints

One common constraint is where the components of $\mathbf{x}$ must be between a lower and an upper bound, $\mathbf{l}$ and $\mathbf{u}$, respectively. These are *box* constraints:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}), \\ \text{s.t.} \quad & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}. \end{aligned} \tag{1.8}$$

It is possible to rewrite these constraints in standard form as:

$$\begin{aligned} \mathbf{l} - \mathbf{x} &\leq \mathbf{0} \\ \mathbf{x} - \mathbf{u} &\leq \mathbf{0}. \end{aligned}$$

Slack Variable Form

It is helpful to know how close the decision variable $\mathbf{x}$ is to the constraint boundaries. For this we introduce a *slack variable* $\mathbf{s} \in \mathbb{R}^m$ and write the constraints in slack variable form:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{s}} \quad & \mathbf{c}^{\mathsf{T}}\mathbf{x}, \\ \text{s.t.} \quad & \mathbf{x}, \mathbf{s} \geq \mathbf{0}, \\ & \mathbf{A}\mathbf{x} + \mathbf{s} = \mathbf{b}. \end{aligned} \tag{1.9}$$

This is equivalent to Eq. (1.4), since $\mathbf{s} = \mathbf{b} - \mathbf{A}\mathbf{x} \geq \mathbf{0}$. Each component s_j of $\mathbf{s}$ quantifies how close $\mathbf{x}$ is to the j-th inequality boundary; $\mathbf{x}$ has as many dimensions as there are constraint equations. This is used in the simplex algorithm, which solves a linear program by iterating through vertices of the feasible region until a local optimum is found.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

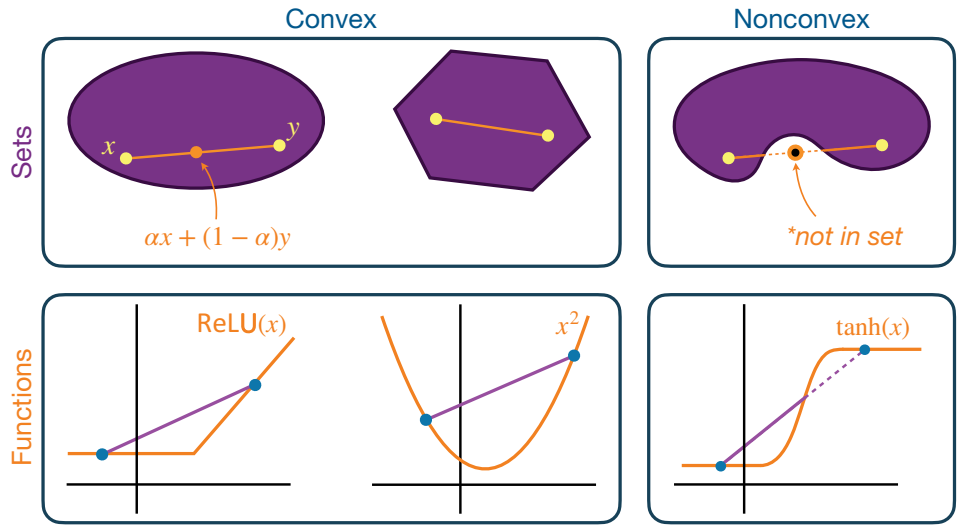


Figure 1.8 Examples of convex and nonconvex sets (top) and convex and nonconvex functions (bottom). Convex sets are defined so that the line segment connecting any two points in the set is itself entirely in the set. Convex functions are defined so that the line segment connecting any two points on the function does not dip below the function at any intermediate values of the domain x. In general, nonconvex feasible sets or nonconvex objective functions complicate the optimization problem considerably. We'll meet the ReLU (rectified linear unit) function in Panel Panel 10.1.

1.2 Convexity in Optimization

Convexity is one of the most important concepts in optimization [66], resulting in strong theoretical guarantees and powerful numerical algorithms. In a convex optimization problem, any local optimal solution is also globally optimal. This is particularly important in high-dimensional optimization problems, where searching for global solutions may be intractable. In contrast, in convex optimization, if an algorithm converges to a local minima, we have confidence that this is the global minima as well.

Convex optimization problems are problems where the objective function is convex and the feasible set determined by the constraint equations are also convex [66, 365, 366]. Occasionally these problems have analytic solutions, such as the classic least-squares regression (see Chapter 4). Other problems, such as linear programming (see Chapter 3), have mature algorithms with favorable computational scaling properties.

In this section, we introduce key notions of convex sets and functions, along with some fundamental theoretical results. Examples of convex sets and functions, as well as nonconvex sets and functions, are shown in Fig. 1.8. The categorization of optimization problems into convex and nonconvex is as fundamental as the categorization of differential equations into linear and nonlinear. Convex optimization problems are largely solved, at least algorithmically, with efficient, robust, and scalable numerical methods.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

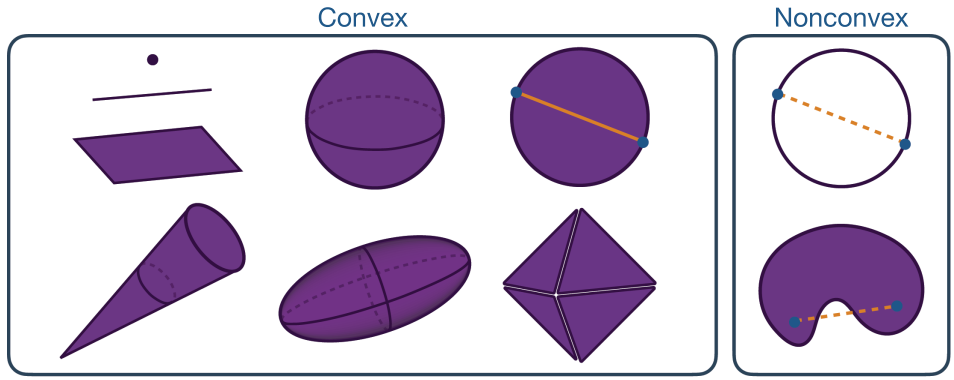


Figure 1.9 Examples of convex and nonconvex sets. Points, lines and line segments, planes and half-planes, spaces and half-spaces (called *slabs*) are all convex. A filled disk or higher-dimensional ball (e.g., a filled sphere), and filled ellipsoids are convex. Polytopes defined by the intersection of multiple inequality constraints generalize polygons to higher dimensions and are also convex. A hollow sphere or a circle without its interior is not convex, because the lines connecting points on the boundary include points that are not in the set; likewise, an object including part but not all of its boundary may be nonconvex.

Convex Sets

It is important to be able to easily assess whether or not the feasible set, defined by the constraint equations, is convex. A set $\boldsymbol{C} \subseteq \mathbb{R}^n$ is convex if and only if the line connecting any two points $\mathbf{x}, \mathbf{y} \in \boldsymbol{C}$ is also entirely in $\boldsymbol{C}$:

$$\alpha\mathbf{x} + (1-\alpha)\mathbf{y} \in \boldsymbol{C} \quad \text{for} \quad \alpha \in [0,1]. \tag{1.10}$$

The parameter $\alpha \in [0,1]$ defines the line segment connecting the two points $\mathbf{x}$ and $\mathbf{y}$ in the convex set $\boldsymbol{C}$; $\alpha = 0$ corresponds to the point $\mathbf{y}$, $\alpha = 1$ corresponds to the point $\mathbf{x}$, and $\alpha \in [0,1]$ sweeps out the line segment between $\mathbf{x}$ and $\mathbf{y}$. This is shown schematically in the top row of Fig. 1.8. If any of these points $\alpha\mathbf{x} + (1-\alpha)\mathbf{y}$ are not also in $\boldsymbol{C}$, for any $\mathbf{x}, \mathbf{y} \in \boldsymbol{C}$, then the set is not convex. In this book, we assume that convex sets are defined in the context of a vector space, although there are more general notions.

Intuitively, a set is convex if there is a continuous *line of site* between any two points in the set, so that there are no disconnected regions, holes, or fingers. It is extremely difficult to search over multiple disconnected regions[5] or fingers, especially in higher dimensions, where the number of such regions may grow exponentially.

Additional examples of convex and nonconvex sets are provided in Fig. 1.9. There are broad classes of convex sets, including points, lines and line segments, planes and half-planes, spaces and half-spaces (sometimes referred to as *slabs*). In addition, several geometric shapes also define convex sets. For example, a disk (i.e., a filled in circle with its boundary) and a higher-dimensional ball

$$B_r = \{\mathbf{x} \,|\, \|\mathbf{x}\|_2 \leq r\}$$

[5] Disconnected regions and holes are a subject of *topology*.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

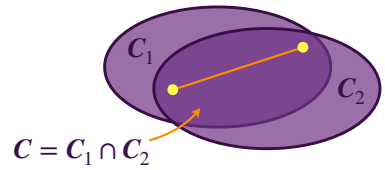


Figure 1.10 The intersection of convex sets is also convex.

(i.e., a filled in sphere) are convex sets. A hollow sphere or a circle without its interior is not a convex set, because the lines connecting points on the boundary are not in the set. Similarly, filled ellipsoids are convex sets. Filled cones, with convex cross-section, are also convex.

Intersection of Convex Sets

The intersection of two convex sets is also convex, as illustrated in Fig. 1.10. This is simple to prove mathematically. Although we won't typically prove results in this book, it is illustrative to see this simple proof.

Proof Denote the two convex sets $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$, and let their intersection be

$$\boldsymbol{C} = \boldsymbol{C}_1 \cap \boldsymbol{C}_2.$$

If $\mathbf{x}, \mathbf{y} \in \boldsymbol{C}$, then $\mathbf{x}, \mathbf{y} \in \boldsymbol{C}_1$ and $\mathbf{x}, \mathbf{y} \in \boldsymbol{C}_2$. Since $\boldsymbol{C}_1$ is convex, then

$$\alpha\mathbf{x} + (1 - \alpha)\mathbf{y} \in \boldsymbol{C}_1$$

for $\alpha \in [0, 1]$. The same is true for $\boldsymbol{C}_2$. Thus,

$$\alpha\mathbf{x} + (1 - \alpha)\mathbf{y} \in \boldsymbol{C}_1 \cap \boldsymbol{C}_2 = \boldsymbol{C}$$

for $\alpha \in [0, 1]$ for all $\mathbf{x}, \mathbf{y} \in \boldsymbol{C}$, which proves the result. □

The intersection of several linear inequality constraints, $\mathbf{A}\mathbf{x} \leq \mathbf{b}$, defines a convex polytope, which generalizes polyhedra to higher-dimensional spaces. An example of a convex polytope is shown in Fig. 1.11. These sets are *pointy*, with the bounding surfaces, edges, and vertices being characterized by subsets of the inequality constraints. Convex polytopes will be explored extensively in the context of linear programming in Chapter 3.

Convex Hull

It is possible to *convexify* a nonconvex set $\boldsymbol{C}$ by systematically addingthe points that violate the convexity condition. The smallest convex set containing a set $\boldsymbol{C}$ is the *convex hull*, and is visualized in Fig. 1.12. This is denoted hull($\boldsymbol{C}$). The convex hull of a convex set $\boldsymbol{C}$ is trivially itself. Further, the convex hull of any set $\boldsymbol{C}$ is the intersection of every convex set containing $\boldsymbol{C}$, hence making it the smallest convex set containing $\boldsymbol{C}$.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

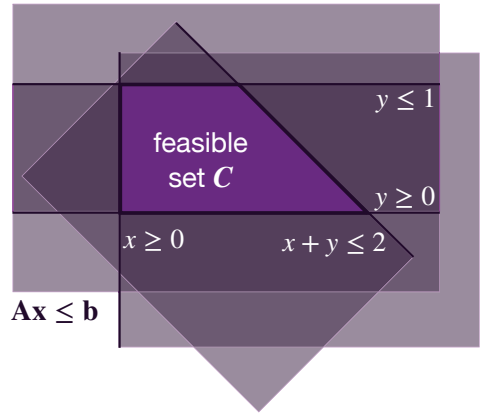


Figure 1.11 Convex polytopes are obtained by the intersection of several linear inequality constraints. Convex polytopes play an important role in several classes of optimization problem, including linear programming (see Chapter 3).



Figure 1.12 Illustration of the convex hull, hull($\mathcal{S}$), of a nonconvex set $\mathcal{S}$.

Affine Sets and Transformations

Affine sets satisfy an even more restrictive condition than convex sets. An affine set is any set in which every line passing through two points of the set lies *completely* within that set. Formally, a set $\mathcal{A}$ is affine if, for any $\mathbf{x}, \mathbf{y} \in \mathcal{A}$ the point $\alpha\mathbf{x} + (1-\alpha)\mathbf{y}$ lies in $\mathcal{A}$ for any real α. In contrast, a *convex* set requires $\alpha\mathbf{x} + (1-\alpha)\mathbf{y}$ to remain in the set *only* for $\alpha \in [0, 1]$. For a convex set, the line segment connecting any two points $\mathbf{x}, \mathbf{y}$ in the set must also be in the set, while for an affine set, the entire infinite line connecting them must be in the set.

Affine sets are automatically convex, because if the points $\alpha\mathbf{x} + (1-\alpha)\mathbf{y}$ are in the set for any scalar α then they are also in the set for $\alpha \in [0, 1]$. A simple example of an affine set is a line $\{\mathbf{x}_0 + t\mathbf{v} : t \in \mathbb{R}\}$. More generally, a plane or hyperplane is affine.

An affine set can be viewed as a "translated" version of a vector subspace. This condition means that $\mathcal{A}$ behaves like a vector space *except* it need not contain the origin, although it may. Thus, an affine set is essentially a vector space that may be shifted or translated by some fixed offset. Affine sets generalize linearvector subspaces by allowing for an arbitrary translation in space. All vector spaces are affine sets, but the converse is not true.

Affine sets arise naturally in optimization when translating or shifting feasible regions. For instance, solving linearsystems $\mathbf{Ax} = \mathbf{b}$ defines an affine subspace. Many linear and affine constraints preserve affine structure, and problems that reduce to finding a point in an affine subspace are simpler than those restricted to convex but curved sets. Hence,

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

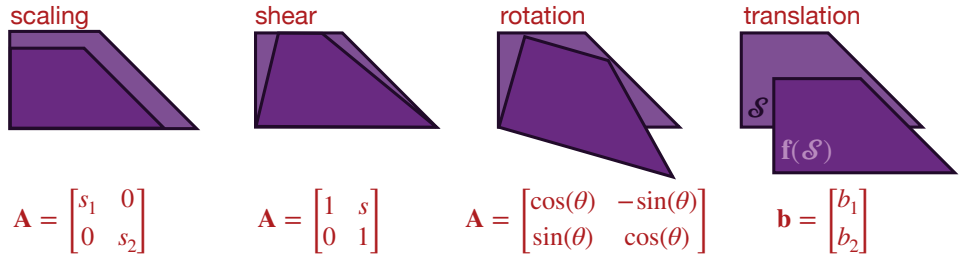


Figure 1.13 Affine transformations $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$. Scaling, shear, rotation, translation, and combinations (such as reflection) thereof, are all affine transformations, which preserve convexity.

in optimization, one often solves feasibility in an affine set, such as $\mathbf{A}\mathbf{x} = \mathbf{b}$, and then superimposes convex inequalities to further restrict that region. Convex sets remain convex under affine transformations, as shown in Fig. 1.13. This is an exercise for the reader.

Panel 1.6: Dual Spaces and Supporting Hyperplanes

Dual spaces and supporting hyperplanes are another important concept in optimization. A supporting hyperplane to a set $C \subset \mathbb{R}^n$ at a boundary point $\mathbf{x}_0$ is a hyperplane

$$\{\mathbf{x} : \mathbf{y}^\top(\mathbf{x} - \mathbf{x}_0) = 0\}$$

such that C lies entirely in one closed half-space determined by that hyperplane. Equivalently, the hyperplane touches C at $\mathbf{x}_0$ but does not cut through its interior. If $C \subset \mathbb{R}^n$ is a convex set, one may define its *dual* set in a dual vector space. Concretely, for a bounded convex body containing the origin, the dual set C^* is given by

$$C^* = \left\{\mathbf{y} \in \mathbb{R}^n : \mathbf{y}^\top\mathbf{x} \leq 1 \ \text{ for all } \mathbf{x} \in C\right\}.$$

Geometrically, each supporting hyperplane to C is described by an equation $\mathbf{y}^\top\mathbf{x} = \alpha$, where $\mathbf{y}$ is a normal. The *dual space* is the collection of all such normal directions $\mathbf{y}$ that define half-spaces containing C; this entirely characterizes the boundary of C.

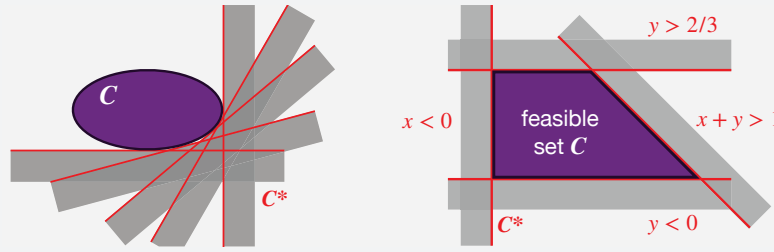


Figure 1.14: Schematic of the dual space of supporting half planes for two convex sets.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

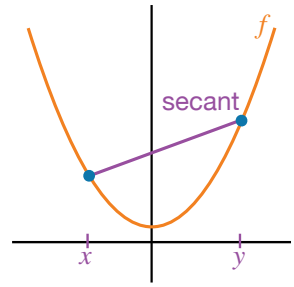


Figure 1.15 A convex function has the property that every secant line connecting any two points on the function, $(\mathbf{x}, f(\mathbf{x}))$ and $(\mathbf{y}, f(\mathbf{y}))$, does not go below the function between $\mathbf{x}$ and $\mathbf{y}$. For a strictly convex function, the secant line is strictly above the function.

Convex Functions

It is also helpful to have conditions to easily assess whether or not an objective function is convex. A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if the line connecting any two points on the function lies above the function:

$$f(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) \leq \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y}). \tag{1.11}$$

This is illustrated in Fig. 1.15. The line connecting the points $\mathbf{x}$ and $\mathbf{y}$ is called the *secant*. Other examples of convex and nonconvex functions are shown in Fig. 1.16. We often use a more restrictive definition, called *strict* convexity, where the inequality in Eq. (1.11) cannot be equal. This prevents degenerate cases where the function has a flat bottom with a line of equal minima. We may also define $f(\mathbf{x})$ for values $\mathbf{x}$ on a restricted domain $\mathbf{x} \in \mathcal{D} \subseteq \mathbb{R}^n$. In this case, f may be considered convex restricted to $\mathcal{D}$ if Eq. (1.11) holds for any $\mathbf{x}, \mathbf{y} \in \mathcal{D}$.

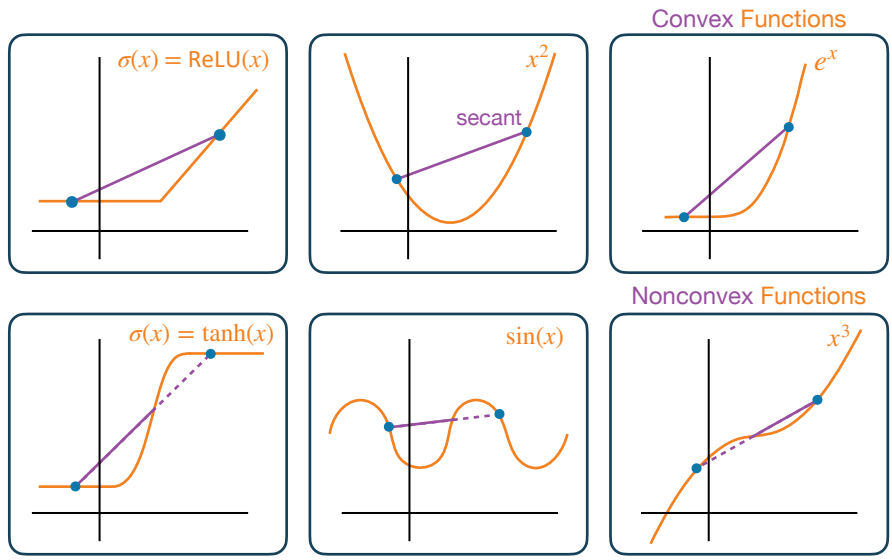


Figure 1.16 Examples of convex and nonconvex functions. The left two functions are commonly used activation functions in the nodes of neural networks, which will be discussed more in Chapter 10.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

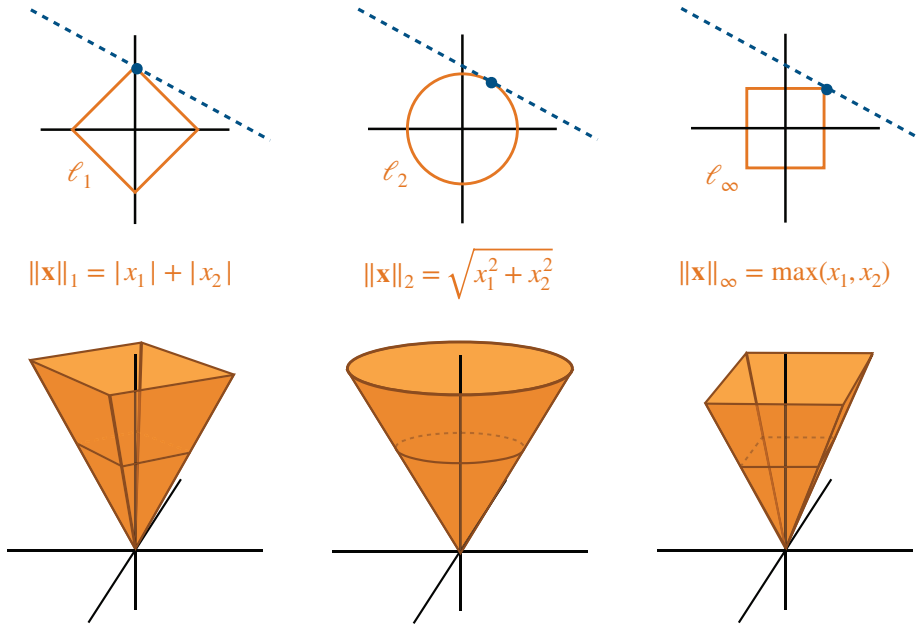


Figure 1.17 Illustration of several norms, all of which are convex functions. This is important in machine learning, where we often minimize the sum-squared error in the ℓ_2 norm, and then add additional regularization terms that are expressed in other norms. The ℓ_1 norm promotes sparsity, the ℓ_2 norm quantifies the sum-squared error, and the ℓ_∞ norm is used to penalize worst-case performance.

Norms are Convex

An important class of convex functions are *norms*, $f(\mathbf{x}) = \|\mathbf{x}\|$. Norms are widely used in optimization, as we often minimize the error of a function fit, where error is quantified in an appropriate norm, such as the ℓ_2 norm. Several norms are shown in Fig. 1.17.

A norm quantifies the length of a vector, and it must satisfy three properties:

1. $\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|$ for all scalar $\alpha \in \mathbb{R}$ (homogeneity)
2. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ (triangle inequality)
3. $\|\mathbf{x}\| \geq 0$ and $\|\mathbf{x}\| = 0$ only if $\mathbf{x} = \mathbf{0}$. (non-negativity)

To prove that all norms are convex, we may write the convexity condition from Eq. (1.11) when $f(\mathbf{x}) = \|\mathbf{x}\|$ as:

$$\begin{aligned}\|\alpha\mathbf{x} + (1-\alpha)\mathbf{y}\| &\leq \|\alpha\mathbf{x}\| + \|(1-\alpha)\mathbf{y}\| \\ &= \alpha\|\mathbf{x}\| + (1-\alpha)\|\mathbf{y}\| \\ &= \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y}).\end{aligned}$$

The most common norms we use are the ℓ_2 and ℓ_1 norms, given by

$$\|\mathbf{x}\|_2 = \left(x_1^2 + x_2^2 + \cdots + x_n^2\right)^{1/2} = \left(\sum_{k=1}^{n} x_k^2\right)^{1/2} \tag{1.12}$$

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

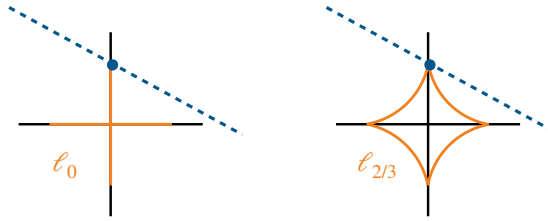


Figure 1.18 Examples of nonconvex *pseudo* norms, which are not actually norms. The ℓ_p norms for $p < 1$ are such nonconvex pseudo norms.

and

$$\|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_n| = \sum_{k=1}^{n} |x_k| \tag{1.13}$$

respectively. The ℓ_2 norm is often used to compute the sum-squared errors of a function fit to data, as in Chapters 4 and 10. The ℓ_1 norm is used to promote solutions that are sparse, meaning that many of the entries of $\mathbf{x}$ are zero.

Both the ℓ_1 and ℓ_2 norms are special cases of the ℓ_p norm for vectors $\mathbf{x}$:

$$\|\mathbf{x}\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{1/p} = \left(\sum_{k=1}^{n} |x_k|^p \right)^{1/p}. \tag{1.14}$$

The gradient of the ℓ_2 norm in two-dimensions, $\|\mathbf{x}\|_2^2 = x_1^2 + x_2^2$, is given by

$$\nabla \|\mathbf{x}\|_2^2 = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} = 2\mathbf{x}.$$

This is useful in the context of least-squares regression (see Chapter 4), where we must minimize the sum of square errors

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 \quad \Longrightarrow \quad \nabla \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 = 0. \tag{1.15}$$

The notion of a convex norm extends to infinite-dimensional L^p Hilbert spaces, where the norm of a function f is now given by

$$\|f\|_p = \left(\int |f|^p \, dx \right)^{\frac{1}{p}}. \tag{1.16}$$

This allows us to extend our optimization framework to function spaces and optimize systems governed by partial differential equations. We now use the Minkowski inequality, which is the triangle inequality for function spaces:

$$\|f + g\|_p \leq \|f\|_p + \|g\|_p.$$

There are several *pseudo* norms that are not actually norms, but are still useful. The ℓ_0 norm (see Fig. 1.18) quantifies how many non-zero entries are in a vector $\mathbf{x}$, which is a measure of its sparsity. Sparsity is essential in inverse problems and machine learning, where we promote models and solutions with the fewest nonzero terms to prevent overfitting. It is

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

possible to minimize the ℓ_0 function, even though it is not convex, by *relaxing* the problem and minimizing the ℓ_1 norm instead. Relaxation is a fundamental technique in nonconvex optimization, and will be discussed more, for example in Chapter 4. Alternatively, it is possible to directly solve the nonconvex ℓ_0 minimization problem using the proximal gradient in Section 5.1.

Regularization and Relaxation

Relaxation and regularization are two important topics in modern optimization that appear throughout the book, starting in Chapter 4. Regularization is the process by which we add terms to the objective function to make ill-posed problems well posed, and it allows one to incorporate prior knowledge and to guide the optimization to solutions with certain properties. For example, we may want to promote sparse or low-rank solutions in many physical contexts. Many regularizing terms also arise as Bayesian priors, as discussed in Chapter 7. However, it is often the case that regularization terms are nonconvex..

It may be possible to relax nonconvex terms in the objective function to related convex approximations, and there is a large body of analysis showing how and when the optimal solutions to the relaxed problems converge to the optimal solutions of the original nonconvex

Panel 1.7: Properties of Convex Functions

There are several important properties of convex functions that we use in this book:

- If $f(\mathbf{x})$ and $g(\mathbf{x})$ are convex, then $f(\mathbf{x}) + g(\mathbf{x})$ is convex. This is useful in the context of machine learning, where we often construct sophisticated loss functions by adding several convex loss terms, as in:

$$L = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \alpha\|\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1.$$

- If $f(\mathbf{x})$ is convex, then $-f(\mathbf{x})$ is concave. The minimum of $f(\mathbf{x})$ becomes the maximum of $-f(\mathbf{x})$.
- If $f(\mathbf{x})$ is convex, then $f(\mathbf{A}\mathbf{x} + \mathbf{b})$ is convex. In other words, an affine transformation of $\mathbf{x}$ preserves convexity.
- If $f(\mathbf{x})$ and $g(\mathbf{x})$ are convex, then the pointwise maximum $h(\mathbf{x})$ of f and g is convex:

$$h(\mathbf{x}) = \max\{f(\mathbf{x}), g(\mathbf{x})\}.$$

- The composition of two functions $h(\mathbf{x}) = f(g(\mathbf{x}))$ is convex if either:
  f is convex and non-decreasing and g is convex;
  f is convex and non-increasing and g is concave.
  For example:
  If g is convex then $\exp(g(x))$ is convex;
  If g is concave then $-\log(-g(x))$ is convex;
  if g is concave and positive, then $1/g$ is convex.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

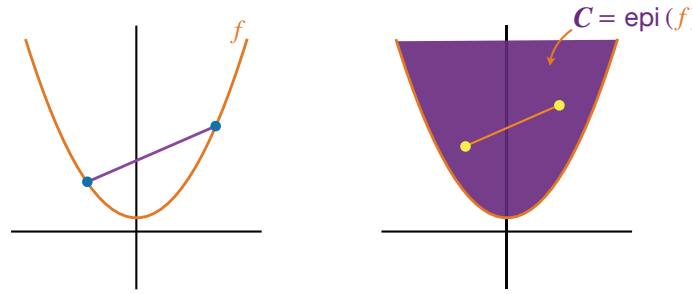


Figure 1.19 The epigraph of a function f is the set of points x, y above the graph. The function f is convex if and only if the epigraph is a convex set.

problems. Often, the addition of a nonconvex term, such as the sparsity promoting $\|\cdot\|_0$ term, leads to an NP-hard optimization problem, where it would require a brute force combinatorial search to find the global optimum. However, it is possible to relax the $\|\cdot\|_0$ pseudo norm by replacing it with the convex $\|\cdot\|_1$ norm. Similarly, if we want a low-rank matrix solution $\mathbf{X}$, we may replace the nonconvex rank $(\mathbf{X})$ with the convex nuclear norm $\|\mathbf{X}\|_*$, which is discussed more in Section 6.5.

Epigraph of a Function

The *graph* of a function is the set of all points on the function:

$$\text{graph}(f) = \{(\mathbf{x}, f(\mathbf{x})) \,|\, \mathbf{x} \in \mathcal{D}\}. \tag{1.17}$$

This is a useful representation of the function as a set of points in the $n+1$-dimensional space $\mathbb{R}^n \times \mathbb{R}$ given by $(\mathbf{x}, f(\mathbf{x}))$.

The *epigraph* of a function is the set of all points above the function:

$$\text{epi}(f) = \{(\mathbf{x}, y) \,|\, \mathbf{x} \in \mathcal{D} \text{ and } y \geq f(\mathbf{x})\}. \tag{1.18}$$

The epigraph of a convex function f is shown in Fig. 1.19. A function is convex if and only if its epigraph is a convex set, which is left as an exercise.

The *hypograph* of a function is the set of all points below the function:

$$\text{hyp}(f) = \{(\mathbf{x}, y) \,|\, \mathbf{x} \in \mathcal{D} \text{ and } y \leq f(\mathbf{x})\}. \tag{1.19}$$

A function is concave if and only if its hypograph is a concave set.

Jensen's Inequality

Jensen's inequality is a powerful and intuitive result for convex functions. In essence, it generalizes the convexity condition for a convex function.

Mathematically, Jensen's inequality may be stated as

$$\alpha_1 f(\mathbf{x}_1) + \alpha_2 f(\mathbf{x}_2) + \cdots + \alpha_m f(\mathbf{x}_m) \geq f(\alpha_1 \mathbf{x}_1 + \alpha_2 \mathbf{x}_2 + \cdots + \alpha_m \mathbf{x}_m) \tag{1.20}$$

for any $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ and $\alpha_j \geq 0$ such that $\alpha_1 + \alpha_2 + \cdots + \alpha_m = 1$.

This inequality has an intuitive geometric interpretation. The standard condition for convexity is that the line segment connecting any two points on the function is not below

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

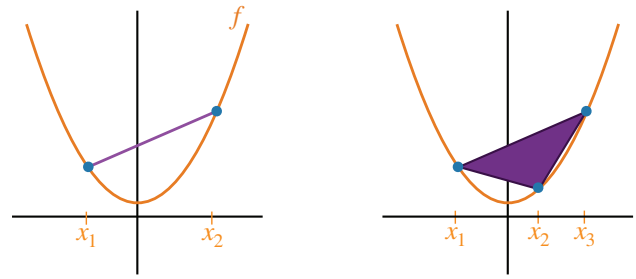


Figure 1.20 Illustration of Jensen's inequality in two-dimensions. Jensen's inequality generalizes the convexity condition that all secant lines lie above the function to state that all convex polytopes formed by m points on the function also lie above the function.

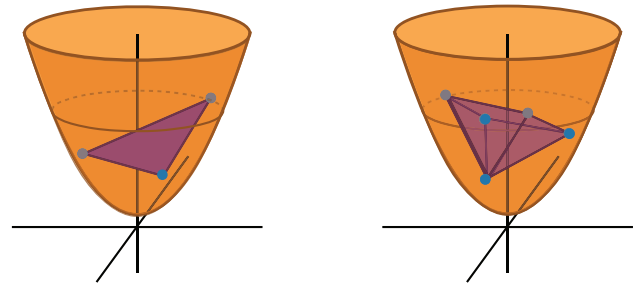


Figure 1.21 Illustration of Jensen's inequality in three-dimensions. In higher dimensions, Jensen's inequality states that any convex polytope formed from m points on the function lie entirely above the function, similar to the secant lines connecting any two points on the function.

the function. Jensen's inequality generalizes this to more than two points on the function. If we have three points (i.e., $m = 3$) on a convex function, the triangle formed with these three points as the vertices also does not lie below the function. This is shown in Fig. 1.20.

This geometric interpretation generalizes to more points and to higher dimensions. Given m points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$, the linear combination $\alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2 + \cdots + \alpha_m\mathbf{x}_m$ forms a convex polytope in $\mathbb{R}^n$. A subset of the points are vertices of this polytope, and other points are interior. Similarly, the points $(\mathbf{x}_1, f(\mathbf{x}_1)), (\mathbf{x}_2, f(\mathbf{x}_2)), \ldots, (\mathbf{x}_m, f(\mathbf{x}_m))$ form a convex polytope that is entirely a subset of the epigraph of f. In other words, this convex polytope lies entirely above the function f, similar to the secant line. This is shown in Fig. 1.21.

Probability Example. A particularly useful application of Jensen's inequality is in probability. We may use Jensen's inequality to prove a fundamental property about the expected value of a function of a random variable x. If $p(x) \geq 0$ and $\int_S p(x)\,dx = 1$, then

$$\underbrace{f\left(\int_S p(x)x\,dx\right)}_{E(x)} \leq \underbrace{\int_S p(x)f(x)\,dx}_{E[f(x)]}. \tag{1.21}$$

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

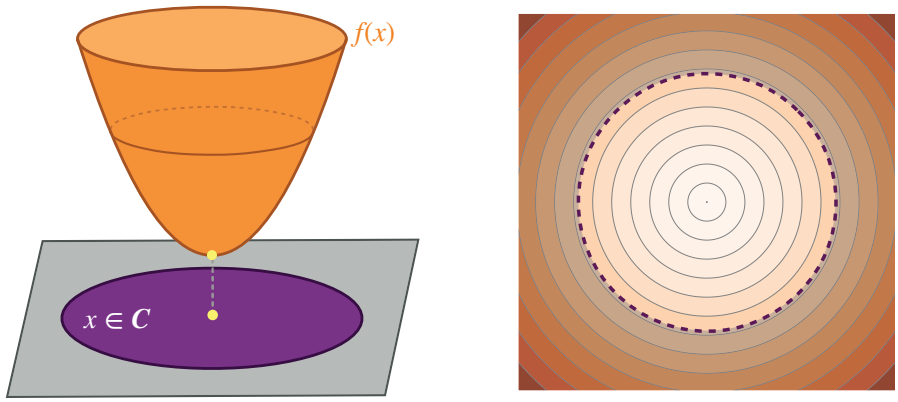


Figure 1.22 Illustration of a convex optimization problem, defined by a convex objective function over a convex feasible set. This example is strictly convex, with a unique global minima.

Thus

$$f\left(E(x)\right) \leq E\left(f(x)\right). \tag{1.22}$$

An example of this inequality is the following:

$$\left(E(x)\right)^2 \leq E\left(x^2\right). \tag{1.23}$$

To prove this using Jensen's inequality, we must extend the finite sum in Eq. (1.20) to a continuous integral. This is left as a homework exercise for the reader.

Convex Optimization Problems

Now we are able to analyze a convex optimization problem. An optimization problem with a convex function f defined over a convex feasible set $\boldsymbol{C}$ is a *convex* optimization problem, shown in Fig. 1.22. If either the function f, or the feasible set $\boldsymbol{C}$ are nonconvex, then the optimization problem is nonconvex as in Fig. 1.23. Nonconvex problems may have multiple local minima, maxima, and saddle points, and the complexity of the optimization landscape typically becomes more challenging in higher dimensions.

We now prove the central result in convex optimization: for a convex function f over a convex set $\boldsymbol{C}$, a local minimum is also a global minimum. We prove this by contradiction.

Proof We assume that $\mathbf{x}$ is a local minimum, so that there is an ϵ-ball around $\mathbf{x}$

$$B_\epsilon(\mathbf{x}) = \{\mathbf{z} \,|\, \|\mathbf{z} - \mathbf{x}\| < \epsilon\}$$

so that any feasible $\mathbf{z} \in B_\epsilon(\mathbf{x})$ satisfies

$$f(\mathbf{z}) \geq f(\mathbf{x}).$$

We now assume that there exists a $\mathbf{y}$ that is a *better* minima

$$f(\mathbf{y}) < f(\mathbf{x}).$$

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

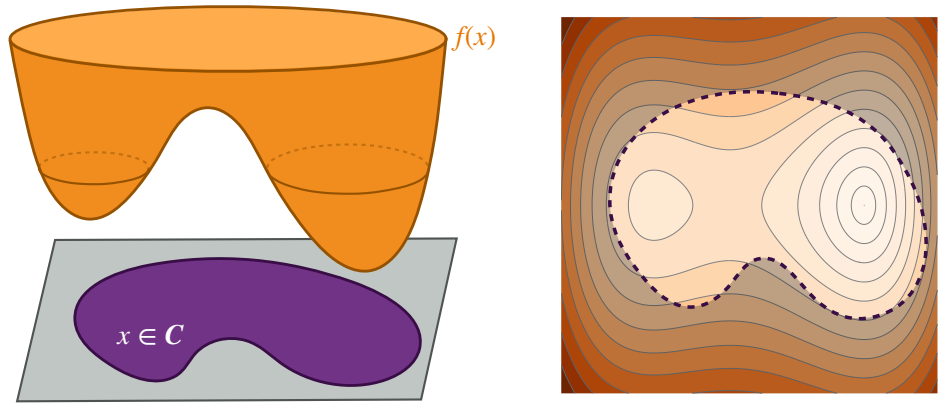


Figure 1.23 Illustration of a nonconvex optimization problem, where the objective function and feasible set are both nonconvex. This example has two local minima and a saddle point.

This is the condition that contradicts the main theorem, and so we will prove that this violates the assumption above.

Because $\mathbf{x}$ and $\mathbf{y}$ are both feasible, in the convex set $\boldsymbol{C}$, then the entire line segment connecting them is also feasible, meaning that it is also in the convex set $\boldsymbol{C}$. These points are parameterized by $\alpha \in [0, 1]$ so that $\alpha = 0$ corresponds to the point $\mathbf{y}$ and $\alpha = 1$ to the point $\mathbf{x}$.

Some of these points are inside the ϵ-Ball. Specifically, we can choose an α so that the point is exactly on the boundary of the ϵ-Ball:

$$\alpha_1 = \frac{\epsilon}{\|\mathbf{y} - \mathbf{x}\|_2}$$

and thus we may choose another α that is halfway between $\mathbf{x}$ and this boundary:

$$\alpha_2 = \frac{\epsilon}{2\|\mathbf{y} - \mathbf{x}\|_2}.$$

We denote this point by $\mathbf{z} = (1 - \alpha_2)\mathbf{x} + \alpha_2\mathbf{y}$.

However, by the convexity of f, and given that $\mathbf{y}$ is a lower minima with $f(\mathbf{y}) < f(\mathbf{x})$, then for every $\alpha \in [0, 1]$, we have

$$\alpha f(\mathbf{y}) + (1 - \alpha) f(\mathbf{x}) < f(\mathbf{x})$$

which implies that

$$f(\mathbf{z}) < f(\mathbf{x}).$$

However, since $\mathbf{z}$ is feasible and $\mathbf{z} \in B_\epsilon(\mathbf{x})$, this contradicts the assumption that $\mathbf{x}$ is a local minimum. □

This is one of the most important results in optimization, providing a powerful guarantee for a large class of convex problems. This property is especially important in Chapters 3 and 4 on linear programming and least squares regression, respectively.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

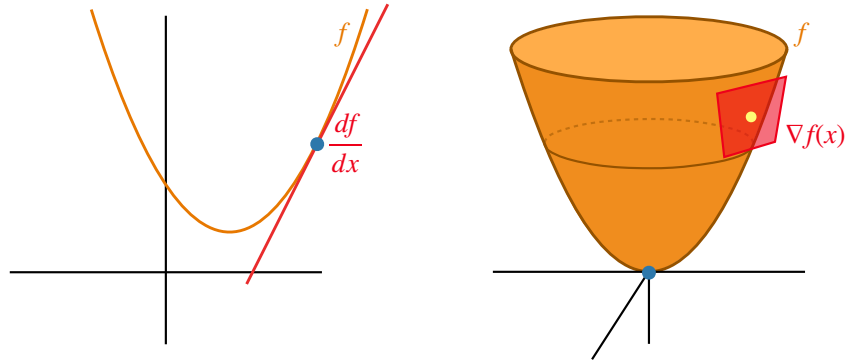


Figure 1.24 Examples of how the derivative and gradient define the tangent line and tangent space, respectively, for a one-dimensional function and a higher-dimensional function.

1.3 Gradients and the Geometry of Optimization

The gradient operator from vector calculus generalizes the derivative of a function of a single variable to the derivative of a function of multiple variables. The gradient has a deep geometric interpretation, quantifying the local direction in which the function increases most rapidly. Thus, we may use this gradient direction to iteratively walk towards a local minimum or maximum of our objective function. This idea is the basis of all gradient descent methods, including the stochastic gradient descent from machine learning, which are discussed in Chapters 2 and 10. The gradientis visualized for a scalar function and a two-dimensional function in Fig. 1.24.

The gradient of a scalar function in two dimensions, $f(x, y)$ is defined as

$$\nabla f = \begin{bmatrix} \partial f/\partial x \\ \partial f/\partial y \end{bmatrix}. \tag{1.24}$$

The components of this vector are partial derivatives that quantify how much the function f changes when x and y are varied infinitesimally. Thus, if the function increases more rapidly in the y direction, the second component will be larger. The gradient of a function $f(\mathbf{x})$ of an n-dimensional vector $\mathbf{x} \in \mathbb{R}^n$ is given by

$$\nabla f = \begin{bmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \\ \vdots \\ \partial f/\partial x_n \end{bmatrix}. \tag{1.25}$$

Each component of ∇f measures how much f changes when the component x_j is varied.

Just as the derivative df/dx of a scalar function $f(x)$ defines a tangent line to the function at the point x, the gradient $\nabla f(\mathbf{x})$ of a function $f(\mathbf{x})$ defines a tangent plane to the function at $\mathbf{x}$. The scalar derivative df/dx is the slope of the tangent line, and $\nabla f(\mathbf{x})$ defines the slope of the tangent plane. Specifically, the gradient $\nabla f(\mathbf{x}_0)$ defines a linear approximation

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

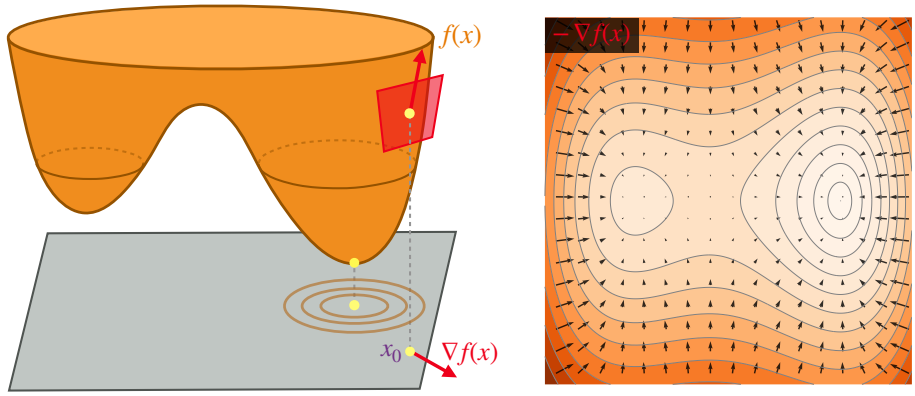


Figure 1.25 Illustration of a nonconvex function, with two local minima. The gradient ∇f is also visualized, both as a tangent plane to the objective function f and as a vector direction over the set of feasible points.

to $f(\mathbf{x})$ near $f(\mathbf{x}_0)$ via a first-order Taylor series approximation:

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^{\mathrm{T}} \underbrace{(\mathbf{x} - \mathbf{x}_0)}_{\Delta\mathbf{x}}. \tag{1.26}$$

The gradient tells how the function f locally changes, given a small change in $\mathbf{x}$. If we denote the small change by $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}_0$, then the change in the function is approximated by $\nabla f(\mathbf{x}_0)^{\mathrm{T}}\Delta\mathbf{x}$. In two dimensions, where $\Delta\mathbf{x} = \begin{bmatrix}\Delta x_1 & \Delta x_2\end{bmatrix}^{\mathrm{T}}$, this can be written approximately as:

$$\nabla f(\mathbf{x}_0)^{\mathrm{T}}\Delta\mathbf{x} = \frac{\partial f}{\partial x_1}\Delta x_1 + \frac{\partial f}{\partial x_2}\Delta x_2. \tag{1.27}$$

One of the most important properties of the gradient in the context of optimization is that it defines the direction in which $\Delta\mathbf{x}$ maximally increases the function f locally. This is easy to see from Eq. (1.26), since $\nabla f(\mathbf{x}_0)^{\mathrm{T}}(\mathbf{x} - \mathbf{x}_0)$ is the inner product of the vector $\nabla f(\mathbf{x}_0)$ and $\Delta\mathbf{x}$, and this inner product is largest when the two vectors point in the same direction.

Similarly, if we choose $\Delta\mathbf{x}$ in the negative gradient direction, $-\nabla f(\mathbf{x}_0)$ causes the function to locally decrease the fastest, providing the basis for gradient descent. The gradient of a nonconvex function is visualized in Fig. 1.25, where it is clear that the negative gradient may be used to locally minimize the function.

In some instances, we might not have access to a gradient, because the objective function might be nonsmooth. For example, the ℓ_1 norm does not have a continuous derivative. In this case, we use the *proximal gradient*, introduced in Section 5.1, to generalize the gradient for nonsmooth functions.

Gradients, Hessians, and Convexity

There are deep connections between the gradient of a differentiable function, $\nabla f(\mathbf{x})$, and convexity. In words, a function is convex if and only if all tangent planes defined by the

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

gradientat every point on the function lie entirely below the function. Thus, the tangent planes are never above the function, and only bound it from below. In fact, they define the supporting hyperplanes for the convex epigraph, as discussed below.

We may write this connection mathematically. A differentiable function $f(\mathbf{x})$ is convex if and only if it lives above its first-order Taylor series approximation, given by

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^{\mathrm{T}} \cdot (\mathbf{x} - \mathbf{x}_0). \tag{1.28}$$

The point $\mathbf{x}_0$ is the base point of the Taylor series expansion, and $\mathbf{x}$ is any other point in the domain of f. This statement is true for all $\mathbf{x}_0$ and $\mathbf{x}$. If there is an $\mathbf{x}^*$ so that $\nabla f(\mathbf{x}^*) = \mathbf{0}$, then $f(\mathbf{x}) \geq f(\mathbf{x}^*)$ for all $\mathbf{x}$. Therefore, $\mathbf{x}^*$ is a global minimum of the function $f(\mathbf{x})$ and $f(\mathbf{x})$ is a convex function. Proving this for a vector $\mathbf{x}$ is not trivial, but it is simpler and can be illustrative to show this for a scalar x.

Second-Order Hessian Condition

Just as the gradient condition in Eq. (1.28) provides a first-order condition for convexity, there is a second-order condition in terms of the Hessian $\mathbf{H} = \nabla^2 f(\mathbf{x})$ for twice continuously

Panel 1.8: Tangent Planes are Supporting Hyperplanes

The graph of $f(\mathbf{x})$ is the pair $(\mathbf{x}, f(\mathbf{x}))$. The tangent plane may be defined by its normal vector, which is perpendicular to the gradient of this point on the graph:

$$\vec{T} = \nabla\left(\mathbf{x}, f(\mathbf{x})\right) = \begin{bmatrix} \mathbf{1} \\ \nabla_{\mathbf{x}} f(\mathbf{x}) \end{bmatrix},$$

where $\mathbf{1}$ is a vector of ones, obtained by taking the gradient of $\mathbf{x}$: $\nabla \mathbf{x} = \mathbf{1}$. The normal vector that is perpendicular to this tangent vector is given by:

$$\vec{n} = \begin{bmatrix} \nabla f(\mathbf{x}) \\ -\mathbf{1} \end{bmatrix}.$$

For a convex f, the set of all tangent planes defines a set of *supporting hyperplanes* that tightly surround the convex set above the function (i.e., the epigraph of f).

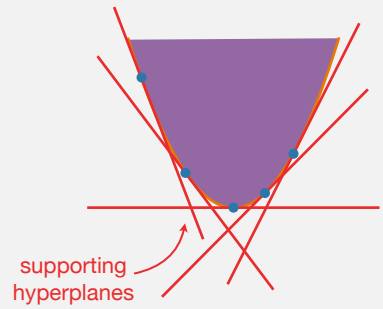


Figure 1.26: Tangents of a convex f define supporting hyperplanes to epi(f).

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

differentiable functions:

$$\mathbf{H} = \nabla^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}. \tag{1.29}$$

A function $f(\mathbf{x})$ is convex if and only if $\nabla^2 f(\mathbf{x}) \geq \mathbf{0}$ everywhere. This can be seen by expanding $f(\mathbf{x})$ in a second-order Taylor series about $\mathbf{x}_0$:

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^{\mathrm{T}}(\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^{\mathrm{T}} \nabla^2 f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0). \tag{1.30}$$

It is clear that Eq. (1.28) is always true if $\nabla^2 f(\mathbf{x}_0)$ is positive semidefinite[6]. When $\mathbf{H} > 0$ is positive definite, then the function is *strictly* convex, meaning that it has a single global minimum; this prevents a line or region of equivalent local minima which is possible when $\mathbf{H} = 0$ and we have a *flat* portion of the objective function. If $\mathbf{H} > 0$ then the function has positive curvature everywhere, meaning it is shaped like a bowl.

Data Fitting Example

Consider the problem of fitting a line to data in Panel 1.3. Recall, the goal is to fit a data set of n points $\{(x_j, y_j)\}_{j=1}^n$ to a straight line of the form $y_j \approx ax_j + b$, where a is the slope and b is the y-intercept. We define the least squares objective as the sum of squared residuals:

$$f(a, b) = \sum_{j=1}^{n} \left(ax_j + b - y_j\right)^2.$$

This is a quadratic function in the parameters a and b, and we can minimize it by taking partial derivatives and setting them equal to zero. Let us compute the gradient:

$$\frac{\partial f}{\partial a} = \sum_{j=1}^{n} 2\left(ax_j + b - y_j\right) x_j,$$

$$\frac{\partial f}{\partial b} = \sum_{j=1}^{n} 2\left(ax_j + b - y_j\right).$$

Setting the gradient equal to zero yields the normal equations, which may also be written as a matrix system of equations:

$$\nabla f(a, b) = 0 \implies \begin{array}{l} \sum_{j=1}^n \left(ax_j + b - y_j\right) x_j = 0, \\ \sum_{j=1}^n \left(ax_j + b - y_j\right) = 0 \end{array} \implies \begin{bmatrix} \sum x_j^2 & \sum x_j \\ \sum x_j & n \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum x_j y_j \\ \sum y_j \end{bmatrix}.$$

Solving this linear system yields the optimal line parameters a^*, b^* that minimize the sum of squared errors. This is the classical least squares solution to linear regression, which we explore more and generalize in Chapter 4.

[6] A matrix $\mathbf{Q}$ is positive semidefinite if $\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} \geq 0$ for all $\mathbf{x}$. We often denote this as $\mathbf{Q} \geq 0$. If $\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} = 0$ only for $\mathbf{x} = 0$ then the matrix is positive definite, denoted as $\mathbf{Q} > 0$.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

1.4 Constraints and Lagrange Multipliers

Constraints are a fundamental ingredient of many optimization problems, and there are several approaches to handle them. The method of Lagrange multipliers provides an elegant way to enforce constraints by embedding them into the objective function. This approach not only offers practical techniques for solving constrained problems but also introduces key ideas in duality, which are central to optimization theory. Lagrange multipliers are widely used in control theory and other engineering applications. Here we provide an intuitive overview of Lagrange multipliers, which are explored more in Chapter 6 and throughout the book.

Consider the simple constrained optimization problem with equality constraints:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}), \\ \text{s.t.} \quad & h(\mathbf{x}) = 0. \end{aligned} \tag{1.31}$$

It is possible to add the constraint equation to the loss function through the use of a penalty function $\delta_0(h(\mathbf{x}))$ which is 0 for $h(\mathbf{x}) = 0$ and ∞ for $h(\mathbf{x}) = 0$:

$$\delta_0(h(\mathbf{x})) = \begin{cases} 0, & h(\mathbf{x}) = 0 \\ \infty, & h(\mathbf{x}) \neq 0. \end{cases} \tag{1.32}$$

If we add this penalty function to the objective function

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) + \delta_0(h(\mathbf{x})) \tag{1.33}$$

then the optimal solution is equivalent to that of the original constrained problem. However, the penalty function is no longer smooth or easy to optimize over. Instead we add $\lambda h(\mathbf{x})$, and as $\lambda \to \pm\infty$, this approaches $\delta_0(h(\mathbf{x}))$; specifically, if $h(\mathbf{x}) > 0$ we take $\lambda \to \infty$ and if $h(\mathbf{x}) < 0$ we take $\lambda \to -\infty$. Technically we don't need to take the limit of λ to plus or minus infinity; at some finite λ, this term becomes large enough to make the constraint active, after which the optimal solution remains the same. Mathematically, we write this as

$$\min_{\mathbf{x}} \max_{\lambda} \quad \underbrace{f(\mathbf{x}) + \lambda h(\mathbf{x})}_{\mathcal{L}(\mathbf{x},\lambda)}. \tag{1.34}$$

The variable λ is a *Lagrange multiplier*, which weights the constraint violation, and the new objective function $\mathcal{L}(\mathbf{x}, \lambda)$ is called the *Lagrangian*. We explore this more in Chapter 6.

The inner optimization operates as follows: for a given $\mathbf{x}$, we maximize the Lagrangian over λ. Again, when the constraint is violated, this maximization is unbounded[7] , since infinite λ results in an infinite Lagrangian. However, when the constraint is satisfied, this term goes to zero, and λ no longer has an affect. Then, the outer optimization searches for $\mathbf{x}$ that minimizes this maximum over λ, which is infinite when the constraint is violated, and is just $f(\mathbf{x})$ when the constraint is satisfied.

[7] Because the maximization problem is unbounded for $h(\mathbf{x}) \neq 0$, this inner $\max_\lambda$ should technically be written as $\sup_\lambda$, the supremum, which is also called the *least upper bound*. This is a technical detail we explore in Chapter 6, but for now we simply write it as max.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

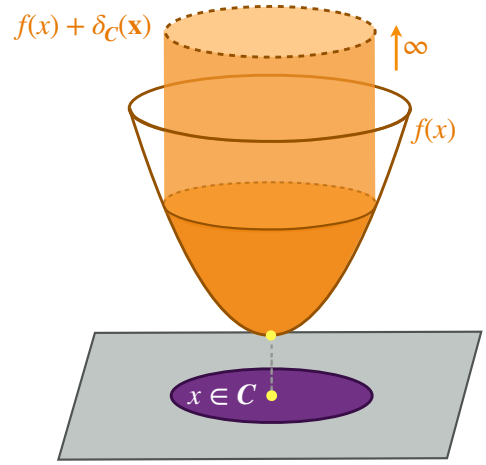


Figure 1.27 Lagrangian objective function with infinite penalty function $\delta_C(\mathbf{x})$ to enforce that $\mathbf{x} \in C$. We will replace this penalty function with Lagrange multipliers.

It is possible to reverse the order of the min and max, resulting in the *dual* optimization problem

$$\max_{\lambda} \min_{\mathbf{x}} \quad \underbrace{f(\mathbf{x}) + \lambda h(\mathbf{x})}_{\mathcal{L}}. \tag{1.35}$$

The optimization problem in Eq. (1.31) is known as the *primal* problem, and Eq. (1.34) is just a different form of the primal. Swapping the order of the $\min_{\mathbf{x}}$ and $\max_{\lambda}$ results in the *dual* problem in Eq. (1.35).

We will show in Chapter 6 that a condition for optimality of these types of constrained problem is that the gradients of the objective function and constraint equation must align:

$$\nabla_{\mathbf{x}} \mathcal{L} = \nabla f(\mathbf{x}^*) + \lambda \nabla h(\mathbf{x}^*) = 0, \quad \nabla_{\lambda} \mathcal{L} = h(\mathbf{x}^*) = 0. \tag{1.36}$$

If there are several equality constraints, so $\mathbf{h}(\mathbf{x}) = 0$, then this may be written as

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{h}(\mathbf{x}). \tag{1.37}$$

Here $\boldsymbol{\lambda}$ is a vector of the Lagrange multipliers.

For inequality constraints, $g(\mathbf{x}) \leq 0$, we modify the notion of the Lagrange multiplier λ to satisfy additional conditions:

$$\lambda \geq 0, \quad \lambda g(\mathbf{x}) = 0. \tag{1.38}$$

This ensures that λ enforces the constraint only when it is active, meaning that the optimal solution is at the boundary of this constraint equation, and not in the interior. This generalized theory of Lagrange multipliers is know as KKT (Karush–Kuhn–Tucker) theory, which is discussed more in Chapter 6.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

For inequality constraints, we use a related penalty function

$$\delta_{\mathbb{R}^+}(g(\mathbf{x})) = \begin{cases} 0, & g(\mathbf{x}) \leq 0 \\ \infty, & g(\mathbf{x}) > 0. \end{cases} \tag{1.39}$$

Again, adding this penalty function to the objective function

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) + \delta_{\mathbb{R}^+}(g(\mathbf{x})) \tag{1.40}$$

is equivalent to the constrained optimization problem. However, the penalty function is no longer smooth or easy to optimize over. Again, we add $\lambda g(\mathbf{x})$ and as $\lambda \to \infty$ this approaches $\delta_{\mathbb{R}^+}(g(\mathbf{x}))$. Similar to above, we arrive at a minimax problem:

$$\min_{\mathbf{x}} \max_{\lambda \geq 0} \quad f(\mathbf{x}) + \lambda g(\mathbf{x}). \tag{1.41}$$

Again, it is possible to reverse the order of the min and sup, resulting in the dual problem[8]:

$$\max_{\lambda \geq 0} \min_{\mathbf{x}} \quad f(\mathbf{x}) + \lambda g(\mathbf{x}). \tag{1.42}$$

The dual provides a lower bound on the primal problem and has practical and theoretical significance, particularly in large-scale optimization and sensitivity analysis. It is not obvious that these formulations are equivalent, but under some conditions, the optimal solutions of each are in fact be equivalent. This notion of duality in optimization will be revisited several times in this book.

Examples of Lagrange Multipliers

Here we explore two examples with specific functions to see how Lagrange multipliers work. The first example involves an active constraint, so that the optimal solution is at the boundary of the constraint, and the second example shows how Lagrange multipliers work when there is a trivial constraint that doesn't change the optimum solution.

Example with Active Constraint

Consider the optimization problem

$$\begin{aligned} \min_{x} \quad & f(x), \\ \text{s.t.} \quad & g(x) \leq 0 \end{aligned}$$

with $f(x) = (x-1)^2 + 1$ and $g(x) = x$. The minimum of the unconstrained optimization problem occurs at $x^* = 1$, but the constraint forces $x \leq 0$. It is possible to plot this objective function and constraint and verify the intuition that the constrained optimal solution is on the boundary of the constraint, so that $x^* = 0$.

Reformulating this problem using Lagrange multipliers yields the dual formulation:

$$\min_{x} \max_{\lambda \geq 0} \; (x-1)^2 + 1 + \lambda x.$$

[8] We discuss duality in much greater detail later, for example in Chapter 6. *Minimax* and *maximin* problems often arise in the context of dual problems, where we minimize over one variable, such as the primal variable $\mathbf{x}$, and maximize over another variable, such as the dual variable $\boldsymbol{\lambda}$.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

Because of convexity, we may swap the order of the inner and outer minimization and maximization and have guarantees that the dual optimum is also the primal optimum. We first compute the inner minimization, which results in

$$\min_x \; f + \lambda g \quad \Longrightarrow \quad \nabla f + \lambda \nabla g = 2x - 2 + \lambda = 0 \quad \Longrightarrow \quad x^* = \frac{1}{2}(2 - \lambda).$$

Next we substitute this into $f(x) + \lambda g(x)$ and maximize with respect to $\lambda \geq 0$:

$$\begin{aligned}\max_{\lambda \geq 0} f(x^*) + \lambda g(x^*) &= \max_{\lambda \geq 0} \left(1 - \frac{\lambda}{2} - 1\right)^2 + 1 + \lambda \left(1 - \frac{\lambda}{2}\right) \\ &= \max_{\lambda \geq 0} \frac{\lambda^2}{4} + 1 + \lambda - \frac{\lambda^2}{2} = \max_{\lambda \geq 0} -\frac{\lambda^2}{4} + \lambda + 1.\end{aligned}$$

The optimum λ is obtained by taking the derivative with respect to λ and setting equal to zero:

$$-\frac{\lambda}{2} + 1 = 0 \quad \Longrightarrow \quad \lambda^* = 2.$$

Plugging this back into our expression for x^* yields $x^* = 0$.

A fundamental question arises: why is $\lambda = 2$ special? Remember, we are not really trying to minimize $f(x) + \lambda g(x)$ but rather minimize $f(x)$ such that $g(x) \leq 0$. If $\lambda < 2$ the constraint is not satisfied. If $\lambda > 2$ the constraint is satisfied but $f(x)$ is not minimized. To visualize this, plot the Lagrangian function for several values of λ, along with the constraint region. This is explored more in Exercise 1.11 and in Chapter 5.

Example with Trivial Constraint

Let's now explore a case where the constraint is not active in the minimization problem. We consider the same constrained optimization problem but with $f(x) = (x + 1)^2 + 1$ and $g(x) = x$. The minimum of the unconstrained problem occurs at $x^* = -1$ which also satisfies the constraint.

Again, we reformulate this problem using the Lagrangian form:

$$\min_x \max_{\lambda \geq 0} \; (x + 1)^2 + 1 + \lambda x.$$

We swap the order of the min and max and solve the inner minimization over x:

$$\frac{\partial \mathcal{L}}{\partial x} = 2x + 2 + \lambda = 0 \quad \Longrightarrow \quad x = -1 - \frac{\lambda}{2}.$$

Substituting this into the Lagrangian yields

$$-\frac{\lambda^2}{4} - \lambda + 1.$$

Maximizing with respect to yields

$$\frac{\partial \mathcal{L}}{\partial \lambda} = -\frac{\lambda}{2} - 2 = 0 \quad \Longrightarrow \quad \lambda = -2.$$

Since this is negative, the maximum of the function for $\lambda \geq 0$ occurs when $\lambda^* = 0$, at which value $x^* = -1$. Thus, when the constraint is not active, then $\lambda^* = 0$.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

1.5 Applications of Optimization

The power of optimization is particularly evident in its many modern applications. Entire books could be written about individual application areas, for example in industrial optimization or network flow optimization.

In this text, we focus on three major application areas to motivate and develop theory: inverse problems (Chapter 8), control theory (Chapter 9), and machine learning (Chapter 10). These applications are of particular modern relevance and represent three frontiers of optimization research. In addition, each of these application domains often involve interacting with or modeling the real, physical world, making them crucial for science and engineering.

Modern applications of optimization are typically high-dimensional and nonlinear, leading to nonconvex problems. There are several approaches to either relax these into convex problems or to tackle the full nonconvex optimization, which we develop in later chapters.

Inverse Problems

Inverse problems seek to infer unknown parameters or inputs that best match observed data, which is typically represented by an equation or system where the solution minimizes the discrepancy between observed and modeled data. For example, seismic inversion is a challenging inverse problem, where the goal is to estimate the geophysical properties of the Earth to best fit seismic data. Thus the goal is to find the unknown parameters $\mathbf{x}$ of a forward model $\mathbf{F}(\mathbf{x})$ that best matched observed data $\mathbf{y}_{\text{obs}}$:

$$\mathbf{F}(\mathbf{x}) \approx \mathbf{y}_{\text{obs}}. \tag{1.43}$$

We typically employ optimization to reframe the problem:

$$\min_{\mathbf{x}} \|\mathbf{F}(\mathbf{x}) - \mathbf{y}_{\text{obs}}\|^2. \tag{1.44}$$

The objective function is the model mismatch $f(\mathbf{x}) = \|\mathbf{F}(\mathbf{x}) - \mathbf{y}_{\text{obs}}\|^2$.

Often, the observations are partial and noisy, and thus don't contain enough information to uniquely determine the unknown parameters, making the inverse problem *ill-posed* or *ill-conditioned*. In such cases, we employ relaxation, regularization, and other prior information to make the problem well posed.

Chapter 8 provides a deep dive into applied optimization for inverse problems. Some of the main examples and case studies include:

Classical inverse problems. There is a rich history of classical inverse problems, such as finding the roots of a polynomial, solving a (potentially ill-posed) matrix system of equations, or estimating the derivative of noisy data. These all provide historical context for fundamental algorithms that are ubiquitous in machine learning and data science.

Inverse problems in medical imaging. Much of the modern work in inverse optimization has centered around inverse problems in imaging sciences, especially in medical imaging. This field is characterized by partial and indirect measurements that

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

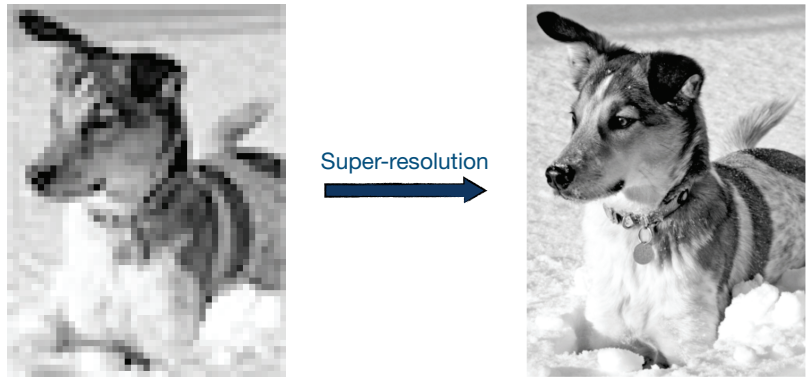


Figure 1.28 Inverse problems, such as super-resolution inversion, are optimization problems, as we seek an inverse solution that minimizes some appropriate loss function. The super-resolution task of inverting a low-resolution image of Mordecai the snow dog to uncover the true high-resolution image is ill-posed, meaning that there are many high-resolution images that yield the same low-resolution input. Thus regularization and other prior information must be added to the optimization problem to promote a unique and reasonable solution.

must be used to reconstruct an image of interest. X-ray computed tomography (CT) and magnetic resonance imaging (MRI) both involve computationally intensive optimization problems, using the Radon transform and Fourier transform, respectively.

Modern computational imaging. Increasingly sophisticated tasks are becoming possible in computational imaging, that were once the subject of science fiction. Figure 1.28 provides one such example of an image-based inverse problem: the task of super resolution, or reconstructing a high-resolution image from a low-resolution image. Other examples include de-noising, de-blurring, filling in damage or occlusions, etc. These are all ill-posed or ill-conditioned inverse problems.

Image-based inverse problem with physics. If an image is obtained from a physical system, it is often possible to use physical priors to regularize the optimization problem. For example, modern computational imaging of black holes involves inverse optimization blended with physics.

PDE-constrained inversion. One of the most challenging classes of inverse problems are those that involve a physical system governed by complex partial differential equation dynamics. For example, designing the shape of a wing to optimize aerodynamic performance, is one of the computational grand challenge problems in engineering. These problems are so challenging in part because of the computational overhead associated with simulating the multiscale dynamics of these systems. Other relevant problems include seismic inversion, or localizing earthquakes and estimating geophysical parameters from seismic data.

System identification. Modeling time series data with ordinary differential equations is a broad topic with a rich history in inverse optimization. This field is closely related to control theory and has dramatically advanced in recent years with advances in applied optimization.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

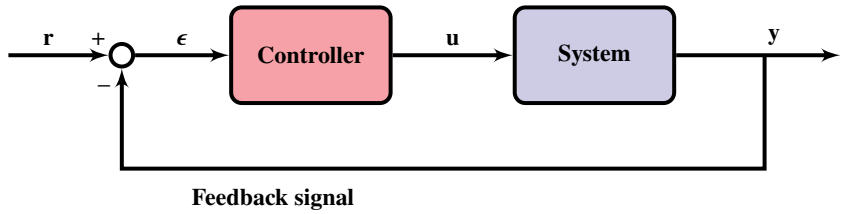


Figure 1.29 Closed-loop feedback control diagram. The sensor signal $\mathbf{y}$ is fed back and subtracted from the reference signal $\mathbf{r}$, providing information about how the system is responding to actuation and external disturbances. The controller uses the resulting error $\boldsymbol{\varepsilon}$ to determine the correct actuation signal $\mathbf{u}$ for the desired response. Feedback is often able to stabilize unstable dynamics while also effectively rejecting disturbances and attenuating sensor noise.

Control Theory

Feedback control is one of the most ubiquitous and important technologies in the modern industrial landscape. The goal of a control system is to design a control law

$$\mathbf{u} = \mathbf{k}(\mathbf{x}) \tag{1.45}$$

that minimizes a cost function

$$J \triangleq J(\mathbf{x}, \mathbf{u}), \tag{1.46}$$

subject to the constraints of a dynamical system that governs the evolution of the system:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{u}). \tag{1.47}$$

The cost function J is often a positive-semidefinite quadratic form, and for linear dynamics, there is an analytic closed form solution, derived using Lagrange multipliers.

In the more general case, we only have access to partial, noisy measurements $\mathbf{y} = \mathbf{g}(\mathbf{x}) + \mathbf{n}$, where $\mathbf{n}$ is the noise. Moreover, the dynamics model is nonlinear and has uncertainty, and there are disturbances to the system. All of these make the optimization problem more challenging.

Much of the technological progress in the past century can be attributed to improvements in our ability to control complex dynamical systems. There are still many challenges in control theory that will undoubtedly benefit from advanced techniques in applied optimization. Chapter 9 explores several topics in optimization for control theory, including:

Linear optimal control. The most complete theory of control has been developed for linear systems, and this topic provides an intuitive mathematical foundation for nearly every other field of control. Linear optimal control includes the linear quadratic regulator (LQR) full-state feedback control (shown in Fig. 1.29) and the Kalman filter, both of which have analytic solutions. The dynamics enters these problems as a constraint, providing an intuitive connection to Lagrange multipliers.

Nonlinear optimal control. Full nonlinear control problems are characterized by the Hamilton–Jacobi–Bellman (HJB) equations, a set of partial differential equations that must be satisfied for an optimal control solution. The HJB equations are based

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

on variation optimal conditions that generalize the classic ideas of variational principles in Lagrangian and Hamiltonian mechanics.

Reinforcement learning. Reinforcement learning (RL) is an important class of optimization problems at the intersection of control theory and machine learning, where the optimal control actions of an agent are learned through a process of exploration and exploitation. RL is a deep topic, and a more modern treatment involving deep learning is provided in the chapter on machine learning.

Dynamic programming. Dynamic programming may be thought of as the optimization engine used to solve nonlinear optimal control problems. DP is also used extensively in reinforcement learning, although modern approaches are increasingly incorporating deep learning.

Model predictive control. Model predictive control (MPC) is perhaps the most widely-used industrial control optimization technique. MPC uses real-time optimization to determine a control action over a receding, finite-time horizon. The performance and applicability of MPC is directly related to our ability to solve challenging optimization problems in real time, and is thus one of the main applications of modern optimization.

Linear matrix inequalities. Linear matrix inequalities (LMIs) are how we solve many linear control problems in practice using modern techniques in optimization. LMIs are often not included in a course on control theory or in a course on optimization. However, they are the backbone of many control implementations.

Differentiable control. If the system model and controller are differentiable, for example represented as neural networks, then powerful gradient-based optimization techniques become available, even for high-dimensional control problems.

Machine Learning

Machine learning is one of the most ubiquitous and compelling modern application of applied optimization. Machine learning involves using optimization to tune the parameters of a model to minimize a loss function averaged over a set of training data; a schematic of a neural network is shown in Fig. 1.30. The machine learning model is a function

$$\mathbf{y} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}) \tag{1.48}$$

where $\mathbf{x}$ are the inputs to the model, $\mathbf{y}$ are the outputs, and $\boldsymbol{\theta}$ are the parameters to be optimized over. The structure of the machine learning model $\mathbf{f}_{\boldsymbol{\theta}}$ defines a space of possible functions that might describe the data, parameterized by $\boldsymbol{\theta}$. For example, $\boldsymbol{\theta}$ might be the parameters of a deep neural network, making this a high-dimensional optimization problem.

In machine learning, the objective function to minimize is called a *loss function* L. The loss function typically includes several terms, with the primary term being the mean squared error (MSE)

$$L = \sum_{j=1}^{N} \|\mathbf{y}_j - \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_j)\|^2 \tag{1.49}$$

summed over the number of data points N.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

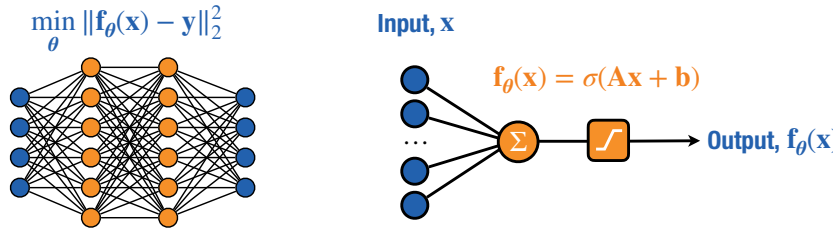


Figure 1.30 Training a neural network is inherently an optimization problem, involving tuning the weights of a neural network to minimize a loss function averaged over the training data.

Chapter 10 provides an in-depth discussion of how optimization is used in various aspects of machine learning, including:

Automatic Differentiation and Backpropagation. The algorithmic evaluation of the derivatives of complex functions is at the heart of optimization for machine learning. These so-called automatic differentiation techniques have been around since the 1960s, but have more recently been used to train neural networks using the backpropagation algorithm. Backpropagation is used to estimate the gradient of the loss function L with respect to the tunable parameters $\boldsymbol{\theta}$, so that gradient descent, and its stochastic variants, may be applied.

Deep learning. Deep learning is the fastest growing field of machine learning, where a deep neural network is used for function approximation. This field is fueled by an abundance of data, modern high-performance computing architectures, and advanced optimization techniques. Stochastic gradient descent is the optimization engine underlying deep learning. It scales to very large systems and incorporates momentum methods, such as the ADAM optimizer, to improve convergence.

Hyperparameter optimization. Machine learning algorithms involve several hyperparameters, such as the size of the neural network, weights of regularizers in the objective, etc. Tuning hyperparameters is a challenging meta-optimization problem.

Deep reinforcement learning. Deep learning has massively improved the scope and applicability of reinforcement learning, for example to improve protein folding, nuclear fusion, to dominate humans in nearly all games, and for other advanced control applications.

Generative methods. Generative AI, such as large language models and diffusion models for image generation, are becoming increasingly powerful and involve advanced optimization methods to formulate and train these models.

Physics-constrained machine learning. Incorporating prior knowledge, such as known physics, is one of the open frontiers of machine learning research. There are two main approaches that are discussed here: (1) incorporating physical priors as regularizing terms in the loss function (e.g., physics-informed neural networks), and (2) directly enforcing physics as constraints in a constrained optimization problem.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

Suggested Reading

Texts

(1) **Convex Optimization**, by S. Boyd and L. Vandenberghe, 2009 [66].
(2) **Numerical Optimization**, by J. Nocedal and S. J. Wright, 1999 [370].
(3) **Practical Methods of Optimization**, by R. Fletcher, 2000 [174].
(4) **Introductory Lectures on Convex Optimization: A Basic Course**, by Y. Nesterov, 2013 [365].
(5) **Constrained Optimization and Lagrange Multiplier Methods**, by D. P. Bertsekas, 2014 [48].
(6) **Lectures on Convex Optimization**, by Y. Nesterov, 2018 [366].
(7) **Engineering Optimization: Theory and Practice**, by S. S. Rao, 2019 [408].
(8) **Practical Optimization**, by P. E. Gill, M. Walter, and M. H. Wright, 2019 [193].
(9) **Introduction to Applied Optimization**, by U. M. Diwekar, 2020 [143].
(10) **Engineering Design Optimization**, by J. R. R. A. Martins and A. Ning, 2021 [340].

Papers and reviews

(1) **CVXPY: A Python-embedded modeling language for convex optimization**, by S. Diamond and S. Boyd, *Journal of Machine Learning Research*, 2016 [140].

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

Homework

Exercise 1.1 An aircraft must fly 2000 miles. It costs \$1000 per hour to fly. In addition, fuel costs \$5 per gallon, and the rate of fuel consumption is $100 + .001V^2$, where V is the cruising speed, measured in miles per hour.

1. What is the minimum-cost cruising speed V, time of flight t, and trip cost C?
2. Suppose the cost of fuel doubles. What is the minimum-cost cruising speed, time of flight, and trip cost?
3. Solve this optimization problem as a function of the fuel cost and plot V, t, and C versus fuel cost.

Exercise 1.2 Consider the projectile problem in Panel 1.1. Imagine we want to hit a particular point at some distance d away. Write this as an optimization problem, and discuss how it might be solved using some of the techniques from this book.

Exercise 1.3 Consider the problem of finding the optimal point to cross a river in Panel 1.2. Let A have coordinates $(0, 0)$ and B have coordinates (L, H), where the distance H is equally spaced between the first stretch of land, the river, and the second stretch of land, so that each has width W and $H = 3W$. More formally write down the optimization problem to solve for x.
Solve for x using calculus. Now write a short computer program to solve for the best x. Compare your answers.

Exercise 1.4 Consider the following general optimization problem domains. In each, determine what are the decision variables $\mathbf{x}$, what is the objective function $f(\mathbf{x})$ and what are the constraints on the system. Try to add as much detail as possible to put these systems into a standard form.

1. Portfolio management in a hedge fund.
2. Optimizing the shape of a wing for aerodynamic performance.
3. Optimizing the pricing of concert tickets in the Sphere.

Exercise 1.5 Determine if the following functions are convex or not. What is the largest domain of convexity around the origin?

1. $f(x) = x^4$.
2. $f(x) = x^4 - .01x^2$.
3. $f(x, y) = -e^{x^2+y^2}$.
4. $f(\mathbf{x}) = -e^{\|\mathbf{x}\|}$.
5. $f(\mathbf{x}) = \sum_j |x_j|$.
6. $f(x, y) = 4\cos(x)^2 + 2\sin(y)^2$.
7. $f(x, y) = 4\cos(x)^2 - 2\sin(y)^2$.
8. $f(x, y) = xy$.

Exercise 1.6 Sketch the feasible sets for the following constraint equations:

1. $S_1 = \{(x, y) \in \mathbb{R}^2 : x \geq 0, y \geq 0\}$;

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

2. $S_2 = \{(x, y) \in \mathbb{R}^2 : y = 2x + 3\}$;
3. $S_3 = \{(x, y) \in \mathbb{R}^2 : y \geq x^2\}$;
4. $S_4 = \{(x, y) \in \mathbb{R}^2 : y^2 \leq x\}$;
5. $S_5 = \{(x, y) \in \mathbb{R}^2 : |x| + |y| \leq 1\}$;
6. $S_6 = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 = 1\}$;
7. $S_7 = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\}$;
8. $S_8 = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \geq 1\}$;
9. $S_9 = \{(x, y) \in \mathbb{R}^2 : 1 \leq x^2 + y^2 \leq 4\}$;
10. $S_{10} = \{(x, y) \in \mathbb{R}^2 : x^2 - y^2 \leq 1\}$.

Which of these are convex or nonconvex?

Exercise 1.7 Prove the following:

1. If $f(\mathbf{x})$ is convex, then $f(\mathbf{x}) + k$ is convex for any constant $k \in \mathbb{R}$;
2. If $f(\mathbf{x})$ is convex, then $\alpha f(\mathbf{x})$ is convex for any constant $\alpha \in \mathbb{R}$;
3. If $f(\mathbf{x})$ is convex, then $f(\mathbf{Ax})$ is convex;
4. If $f(\mathbf{x})$ is convex, then $f(\mathbf{x} + \mathbf{b})$ is convex;
5. If $f(\mathbf{x})$ is convex, then $f(\mathbf{Ax} + \mathbf{b})$ is convex for any affine transformation $\mathbf{Ax} + \mathbf{b}$. If the original domain is $\mathcal{D}$, what is the new domain?

Exercise 1.8 Show that the triangle inequality holds for the following four norms:

1. $f(\mathbf{x}) = \|\mathbf{x}\|_1$;
2. $f(\mathbf{x}) = \|\mathbf{x}\|_2$;
3. $f(\mathbf{x}) = \|\mathbf{x}\|_2^2$;
4. $f(\mathbf{x}) = \|\mathbf{x}\|_\infty$.

Exercise 1.9 Compute the gradient of the four norms in the previous problem in two-dimensions and visualize in the x_1-x_2 plane.

Exercise 1.10 Compute the gradient ∇f and Hessian $\nabla^2 f$ of the following functions:

1. $f(x, y) = x^2 + y^2$;
2. $f(x, y) = x^4 - x^2y^2 + y^4$;
3. $f(x, y) = x^2y + y^2x$;
4. $f(x, y) = e^{x^2+y^2}$;
5. $f(x, y) = \sin(xy)$;
6. $f(\mathbf{x}) = \|\mathbf{x}\|^2 = \mathbf{x}^\mathrm{T}\mathbf{x}$;
7. $f(\mathbf{x}) = \|\mathbf{x}\|_\mathbf{Q}^2 = \mathbf{x}^\mathrm{T}\mathbf{Q}\mathbf{x}$;
8. $f(\mathbf{x}) = \mathbf{a}^\mathrm{T}\mathbf{x} + b$, where $\mathbf{a} \in \mathbb{R}^n, b \in \mathbb{R}$;
9. $f(\mathbf{x}) = \ln(1 + \|\mathbf{x}\|^2)$;
10. $f(\mathbf{x}) = e^{\mathbf{a}^\mathrm{T}\mathbf{x}}$, where $\mathbf{a} \in \mathbb{R}^n$.

Exercise 1.11 Determine the extrema of the following functions, and which of these are local/global minima:

1. $f(x) = \cos(x)$;

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

2. $f(x) = e^{-x^2}$;
3. $f(x) = -e^{(x-5)^2}$;
4. $f(x, y) = x^2 - y^2$;
5. $f(x, y) = x^4 + y^4 - x^2y^2$;
6. $f(x, y) = x^2 + 2y^2 - 4x$;
7. $f(x) = x^6 - 3x^4 + 2x^2$;
8. $f(\mathbf{x}) = \|\mathbf{x}\|^2 + 2\mathbf{a}^{\mathrm{T}}\mathbf{x} + b$, where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{a}, b$ are constants;
9. $f(\mathbf{x}) = \|\mathbf{x}\|^3$, where $\mathbf{x} \in \mathbb{R}^n$.

Exercise 1.12 Identify the following optimization problems as convex or nonconvex, and justify your answer:

1. Minimize $f(\mathbf{x}) = x_1^2 + x_2^2$ subject to $x_1 + x_2 = 1$;
2. Minimize $f(\mathbf{x}) = e^{x_1} + e^{x_2}$ subject to $\| \mathbf{x}\| \leq 2$;
3. Maximize $f(\mathbf{x}) = x_1x_2$ subject to $x_1, x_2 \geq 0$ and $x_1 + x_2 = 5$.

Exercise 1.13 Show that the L_0 "norm"

$$\|\mathbf{x}\|_0 = \#\text{nonzero entries of } \mathbf{x}$$

is not actually a norm. What property does it violate?

Exercise 1.14 Explain why the ℓ_1 norm is a suitable proxy for the ℓ_0 pseudo norm. Does this relaxation become more or less effective for higher dimensional $\mathbf{x} \in \mathbb{R}^n$?

Exercise 1.15 Prove that a function is convex if and only if its epigraph is convex.

Exercise 1.16 Prove that for three points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ then $\alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2 + \alpha_3\mathbf{x}_3$ forms a convex triangle in $\mathbb{R}^n$ if $\alpha_1 + \alpha_2 + \alpha_3 = 1$. This generalizes Jensen's inequality.

Exercise 1.17 Prove that for a set of m points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ then $\alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2 + \cdots + \alpha_m\mathbf{x}_m$ forms a convex polytope in $\mathbb{R}^n$ if $\alpha_1 + \alpha_2 + \cdots + \alpha_m = 1$. Give an example where every point $\mathbf{x}_j$ is a vertex of the polytope, and give an example where some of the points are not vertices of the resulting polytope.

Exercise 1.18 Prove Jensen's inequality.

Exercise 1.19 Generalize Jensen's inequality to continuous integrals, as in Eq. (1.21).

Exercise 1.20 **[Hard]** Show that if a set $\mathcal{S} \subseteq \mathbb{R}^n$ is convex, then its projection onto any subset of of the coordinates of $\mathbb{R}^n$ is also convex.

Exercise 1.21 **[Hard]** Show that if a set $\mathcal{S} \subseteq \mathbb{R}^n$ is convex, then its affine projection onto any plane will also be convex.

Exercise 1.22 Show that if a set $\mathcal{S} \subseteq \mathbb{R}^n$ is convex, then its projection onto any affine space is convex.

Exercise 1.23 Given two points $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, show that the set of points closer to $\mathbf{a}$ than $\mathbf{b}$ is a convex half-plane:

$$\mathcal{S} = \{\mathbf{x} \,|\, \|\mathbf{x} - \mathbf{a}\| \leq \|\mathbf{x} - \mathbf{b}\|\}.$$

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press

Exercise 1.24 (Boyd and Vandenberghe 3.1) Given a convex function $f(x) : \mathbb{R} \to \mathbb{R}$, show that

$$\frac{f(x) - f(a)}{x - a} \leq \frac{f(b) - f(a)}{b - a} \leq \frac{f(b) - f(x)}{b - x}$$

for $x \in (a, b)$. Draw a geometric sketch to demonstrate this.

Exercise 1.25 This exercise will explore the convex properties of matrices.

1. Show that the maximum eigenvalue $\lambda_{\max}(\mathbf{X})$ of a matrix $\mathbf{X}$ is a convex function.
2. Show that the log of the determinant of a matrix, $\log\det\mathbf{X} = \sum_i \log\lambda_i$ is a convex function for symmetric positive definite matrices $\mathbf{X}$.
3. Show that the trace, $\mathrm{Tr}(\mathbf{X}) = \sum_i \lambda_i$ is a linear function, i.e. that $\mathrm{Tr}(\mathbf{X}+\mathbf{Y}) = \mathrm{Tr}(\mathbf{X})+\mathrm{Tr}(\mathbf{Y})$, and hence is a convex function.

Exercise 1.26 Prove that the Cartesian product of two convex sets is a convex set.

Exercise 1.27 Let C_1 and C_2 be convex. Then their Minkowski sum $C_1 + C_2 = \{x + y | x \in C_1, y \in C_2\}$ is also convex. Prove this and give some examples.

Exercise 1.28 Using Lagrange multipliers, solve the problem

$$\min_{x,y\in\mathbb{R}} \quad x^2 + y^2 \quad \text{subject to} \quad (x-1)^2 + (y-1)^2 = 2.$$

Plot the constraint region and identify the minimizing point(s).

Exercise 1.29 In this problem, we will revisit the Lagrange multiplier example from Section 1.4. Plot the Lagrangian form of the objective function

$$(x-1)^2 + 1 + \lambda x$$

for several values of $\lambda > 2$, $\lambda < 2$, and $\lambda = 2$. Include the constraint $x \leq 0$ on the plot. Which value of $\lambda \geq 0$ maximizes the Lagrangian?

Exercise 1.30 For each of the affine transformations $\mathbf{T}$ (scaling, shear, rotation, and translation), show that they preserve the convexity of a set $\boldsymbol{C}$. In particular

$$\mathbf{T}(\boldsymbol{C}) = \{\mathbf{T}\mathbf{x} \,|\, \mathbf{x} \in \boldsymbol{C}\}$$

is convex.

Exercise 1.31 Consider two convex sets in 2D: a closed unit disk and a closed unit square, both centered at the origin. Show by hand how the affine transformations (scaling, shear, rotation, and translation) change these sets. Then write a code to apply these transformations and plot the resulting set.

Exercise 1.32 Show that all affine transformations (scaling, shear, rotation, and translation) are invertible.

https://doi.org/10.1017/9781009755856.002 Published online by Cambridge University Press