Contents

Chapter One

Gradient-Based Methods for Deterministic Continuous Optimization

This chapter presents a summary of salient results in deterministic optimization, particularly focusing on numerical methods. For basic definitions, and results we refer to standard textbooks.

1.1 UNCONSTRAINED OPTIMIZATION

Consider a cost function $J(\theta)$, with $J: \Theta \subseteq \mathbb{R}^d \mapsto \mathbb{R}$, where θ is a decision vector. Throughout this monograph, we seek to the find the minimum of $J(\theta)$ for $\theta \in \Theta$. As is standard in the literature, we are not only interested in the value of the global minimum (if it exists) but also its location, i.e., we seek the solution θ^* to the problem

$$\arg \min_{\theta \in \mathbb{R}^d} J(\theta). \tag{1.1}$$

In the case that the global minimum is attained at several locations, θ^* is one of these locations. In the case that $J(\cdot)$ is an (affine) linear mapping, the above optimization problem is called a *linear problem* and it can be addressed with methods from the theory of linear optimization. See, for example, [84, 85, 224, 235] for details. In the case that $J(\cdot)$ is a general "smooth" continuous real-valued function, the above problem is called a *non-linear problem* and it is referred to as an NLP. The theory presented in this monograph is devoted to the study of NLPs. It is worth noting that while the results presented here can also be applied to linear problems, there are often more efficient methods available for linear problems exploiting the linear nature of the problem.

We assume that $\mathbb{R}^d$ is equipped with a norm denoted by $||\cdot||$. Most results presented in the following are independent of the choice of $||\cdot||$. Occasionally, we will work with the Euclidean norm on $\mathbb{R}^d$ given by

$$\|x\| = \sqrt{x_1^2 + \ldots + x_d^2},$$

and when results only hold for this particular norm it will be stated in the text.

A particular class of applications arises when an input data vector x and corresponding output data vector $h(x)$ is available. Letting $f(\theta, x)$ denote some parametrized mapping proposed for replacing the unknown mapping $h(x)$, considering

$$J(\theta, x) = \|f(\theta, x) - h(x)\|^2$$

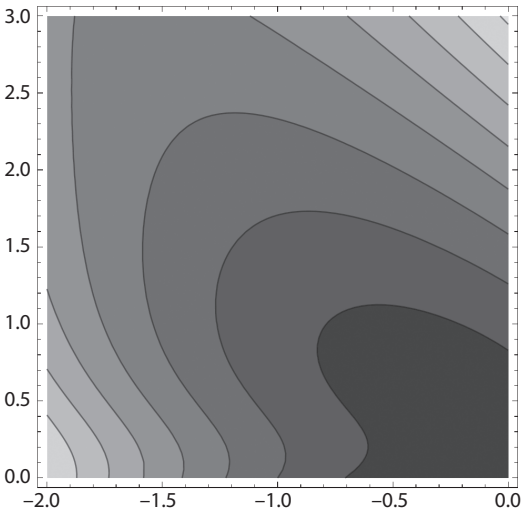


Figure 1.1. Plot showing various level curves for the function $x^2 + (x+2)y + 1/(y^2+0.5)$.

and solving (1.1) for given x, yields then the best fit to the output. This is called *supervised learning* in the literature. In this monograph, we will discuss classical optimization as well as learning applications.

Definition 1.1. The *level sets* of a function $J\colon \mathbb{R}^d \to \mathbb{R}$ are defined for every level $\alpha \in \mathbb{R}$ as:

$$\mathcal{L}_\alpha(J) = \{\theta \in \mathbb{R}^d \colon J(\theta) \le \alpha\}.$$

When no confusion arises, the notation will be simplified to $\mathcal{L}_\alpha$.

Notation. Denote the n-times continuously differentiable mappings from $\mathbb{R}^d$ to $\mathbb{R}$ by C^n. For $J \in C^1$, we denote the gradient of $J(\cdot)$ by $\nabla J(\cdot)$, and for $J \in C^2$, we denote the Hessian of $J(\cdot)$ by $HJ(\cdot) = \nabla^2 J(\cdot)$. Following standard notation, vectors in $\mathbb{R}^d$ are *column* vectors. For $x \in \mathbb{R}^d$, we denote the i-th element of x by x_i. In case of a sequence of vectors $\{x_n\}$, with $x_n \in \mathbb{R}^d$, we denote the i-th element of x by $x_{n,i}$. The gradient is a *row* vector with components $\partial/\partial\theta_k$, $k = 1, \ldots d$. The Hessian is a $d \times d$ matrix with (i, j)-components $\partial^2/\partial\theta_i\partial\theta_j$. For a vector $v \in \mathbb{R}^d$ we write $v \ge 0$ if $v_i \ge 0$ for all components $i = 1, \ldots, d$.

A matrix $B \in \mathbb{R}^{d\times d}$ is negative (positive) definite if $v^\top Bv < (>)0$ for all $v \in \mathbb{R}^d$ with $v \ne 0$, where $v^\top$ denotes the transpose of v. It is called "semi"-definite if the strict inequality equality "<" is replaced by inequality "≤." The notation $B < (>)0$ is often used. A square matrix B is called symmetric if $B = B^\top$. For symmetric matrices the following characterization of positive definiteness exists: if B is symmetric, then $B > 0$ if and only if all its eigenvalues are strictly positive.

Remark 1.1. The visual interpretation of the gradient of a function will be very useful in the rest of this book. Refer to Figure 1.1. This is a "topographical" visualization of a two–dimensional function, where the shades of gray indicate height. Each of the level sets defines a boundary (in the example, they are ellipses). The gradient of the function (in this case, $x^2 + 2y^2$) records the rate of growth of the function along each of the axes. Now, take any point on a level set (refer to Figure 1.2). Because the function does not change *along* this curve, then necessarily the gradient ∇J must point *perpendicular* to the curve (i.e., the

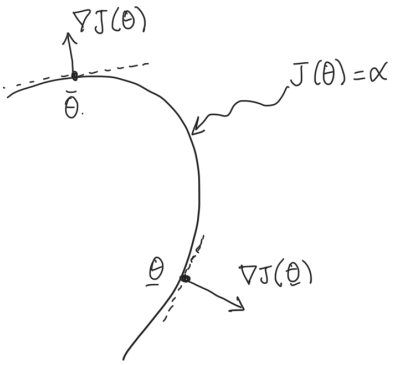


Figure 1.2. Illustration of the gradient of a convex function at two points $\bar{\theta}$ and $\underline{\theta}$.

projection of the gradient on the curve is zero). For the example, it points outward in the direction of growth.

Definition 1.2. A function $J: \mathbb{R}^d \to \mathbb{R}$ is called *concave (convex)* if for all $x, y \in \mathbb{R}^d$ and $\alpha \in (0, 1)$,

$$J(\alpha x + (1-\alpha)y) \geq (\leq)\ \alpha J(x) + (1-\alpha)J(y). \tag{1.2}$$

Strict concavity (convexity) is obtained when the above inequalities are strict. For $J \in C^2$, an equivalent condition is that the Hessian of the function be negative (positive) semi-definite: $\nabla^2 J(\cdot) \leq (\geq)\ 0$, and strict concavity (convexity) follow when the Hessian is negative (positive) definite throughout the domain of J.

Definition 1.3. A point $\theta^* \in \mathbb{R}^d$ can be characterized as follows:

- If $J(\theta^*) \leq J(\theta)$, for all $\theta \in \mathbb{R}^d$, then θ^* is a called *global* minimum. It is called a *local minimum* if there is a $\rho > 0$ such that $||\theta - \theta^*|| \leq \rho$ implies $J(\theta) \geq J(\theta^*)$.
- If $J(\theta^*) \geq J(\theta)$, for all $\theta \in \mathbb{R}^d$, then θ^* is called a *global* maximum. It is called a *local maximum* if there is a $\rho > 0$ such that $||\theta - \theta^*|| \leq \rho$ implies $J(\theta) \leq J(\theta^*)$.
- If the function may increase or decrease in a small neighborhood of the point, depending on the direction of motion, then $\theta^* \in \mathbb{R}^d$ is a *saddle point*.

If, in the definition of a maximum (respectively, minimum), the inequality $J(\theta^*) \leq J(\theta)$ (respectively, $J(\theta^*) \geq J(\theta)$) can be replaced by a strict inequality, then we say that the maximum (respectively, minimum) is *strict*.

Let $\alpha^* \stackrel{\text{def}}{=} \min J(\theta)$, with $\alpha^* = J(\theta^*)$. Then we say that $J(\theta)$ has a proper minimum, and we call α^* the *value of the minimum* and θ^* the *location of the minimum* (the concepts are defined for maxima analogously). Consider the mapping $J(\theta) = e^{-\theta}$ for $\theta \in \mathbb{R}$. For this function we have $0 = \alpha^* = \inf_\theta J(\theta)$ but there exists no value θ so that $J(\theta)$ attains 0. In this case, we will say that $J(\theta)$ has 0 as an improper minimum. Even at this early stage, it is conceivable that any gradient-based search algorithm will run into (numerical) difficulties in the presence of improper minima. In the following, we will only consider proper minima, and we call them minima for short. Whenever appropriate, we will also discuss improper minima, but this will be on an ad hoc basis. Note that the value of a (proper) minimum is unique (provided it exists) but there may be more than one location yielding the same minimal value of $J(\theta)$. In fact the level set of $J(\theta)$ for level α^*, denoted as $\mathcal{L}_{\alpha^*}$, yields the set of all locations of the global minima of $J(\theta)$. Figure 1.3 shows an example with various minima,

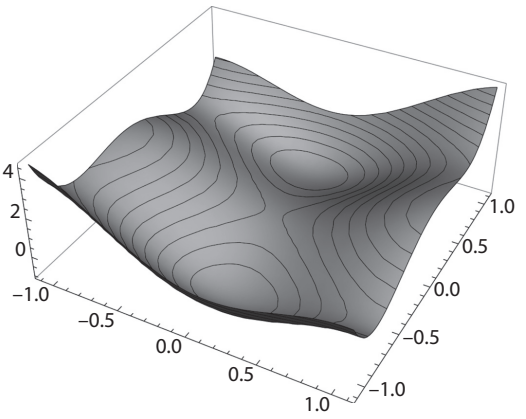


Figure 1.3. Example of a function with several maxima, minima, and saddle points.

maxima, and saddle points. The function is $(4-2.1\theta_1^2+\frac{\theta_1^4}{3})\theta_1^2+\theta_1\theta_2+(-4+4\theta_2^2)^2\theta_2^2$, the so-called six-hump camel back function.

Definition 1.4. The points $\bar{\theta}\in\mathbb{R}^d$ that satisfy $\nabla J(\bar{\theta})=0$ are called *stationary* points of J.

The following theorem provides conditions for deciding the type of a stationary point.

Theorem 1.1. *Let* $J\in C^2$.

- *A local minimum (local maximum)* θ^* *of* J *is a stationary point, that is, it satisfies the* first-order optimality condition*:*

$$\nabla J(\theta^*)=0. \tag{1.3}$$

- *A decision value* θ^* *is a local minimum (local maximum) of* J *if in addition to (1.3), the following is also satisfied*

$$\nabla^2 J(\theta^*)>0 \qquad \left(\nabla^2 J(\theta^*)<0\right). \tag{1.4}$$

  Equation (1.4) is called second-order optimality condition*.*
- *If* J *is a* convex *(concave) function, then (1.3) is necessary and sufficient for* θ^* *being a global minimum (maximum).*

Proof. For a twice continuously differentiable function $J(\theta)$, i.e., $J\in C^2$, the Taylor series expansion with remainder yields the following expression of the value of J at a point $\theta+t\eta$, for $\eta\in\mathbb{R}^d$ and $t\in\mathbb{R}^+$:

$$J(\theta+t\eta)=J(\theta)+t\,\nabla J(\theta)\eta+\frac{t^2}{2}\eta^\top\nabla^2 J(\theta')\,\eta, \tag{1.5}$$

where $\theta'=\theta+t'\,\eta$, for some "intermediate" value $0\le t'\le t$.

Use the above expression around θ^* to express $J(\theta^*+t\eta)$ for an arbitrary direction η. The definition of a positive definite matrix implies that $\eta^\top\nabla^2 J(\theta^*)\eta>0$ for all $\eta\in\mathbb{R}^d$. In addition, $\nabla^2 J(\cdot)$ is continuous, so use a sufficiently small value of t to complete the arguments. The details are left as an exercise. □

Example 1.1. A well-known historical problem is that of explaining the phenomenon of refraction of light when traversing two different media. Since Ptolemy (circa 140 AD),

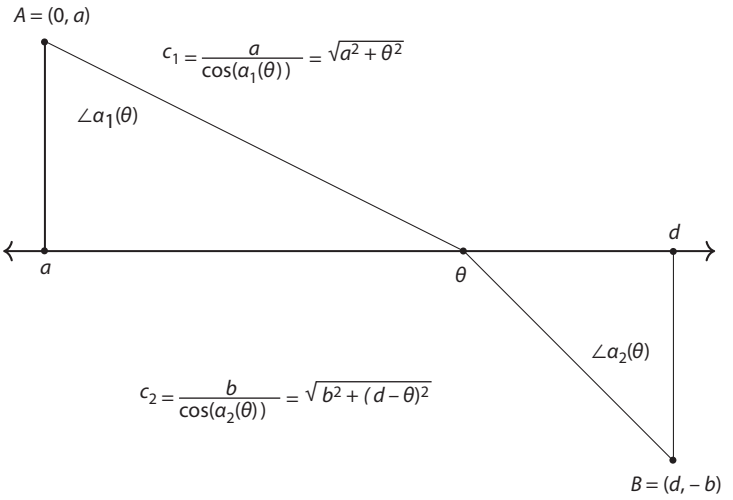


Figure 1.4. Problem is to find the optimal crossing point θ.

scientists were concerned with finding the relationship between the angles of refraction and the media's characteristics. The law of refraction was described by Ibn Sahl of Baghdad (978 AD), who used it to shape lenses, and in 1621 by the Dutch astronomer Snellius. It is now called Snell's Law in English. In 1637 Descartes found the same principle using conservation of moments, and in French it is called the Law of Snell-Descartes. We are mostly interested in Fermat, who in 1657 used variational calculus and his principle of "least time" to derive this law through an optimization problem.

The version of the problem that we give here is the pedagogical version of Richard Feynman. Imagine that you are walking on the beach when you see a person drowning and shouting for help. To get from where you are to the drowning person in the fastest way, you should not move along the straight line, because you run faster on the sand than you can swim. The distance from your position to the water is a, the distance from the water to the person is b, and the length of shoreline between the two points is d, as shown in Figure 1.4.

In Cartesian coordinates the drowning person is at $B = (d, -b)$ and your position is $A = (0, a)$. Here we assume that the waterfront is a straight line for simplicity. The speed on sand is v_1 and in the water v_2, with $v_2 < v_1$. We call θ the crossing point.

Fermat reasoned that light chooses not the shortest path but the one that saves more energy, which is the *fastest* path. It is easy to argue the existence of a solution for this problem: any value of θ to the left of the crossing point of the straight line between points A and B will give a slower path than the straight line because of $v_2 < v_1$. On the other hand, any point $(0, \theta)$, with $\theta > d$, will require unnecessary additional travel time compared to crossing at $(0, d)$, therefore there must be a minimum point between these two points. That the travel times are continuously differentiable follows from the linear relationships between distance and time.

At speed v, the distance traveled in time t is vt, so the total travel time can be expressed as

$$\begin{aligned} J(\theta) &= \frac{1}{v_1} \frac{a}{\cos(\alpha_1(\theta))} + \frac{1}{v_2} \frac{b}{\cos(\alpha_2(\theta))} \\ &= \frac{a}{v_1} \sec(\alpha_1(\theta)) + \frac{b}{v_2} \sec(\alpha_2(\theta)), \end{aligned}$$

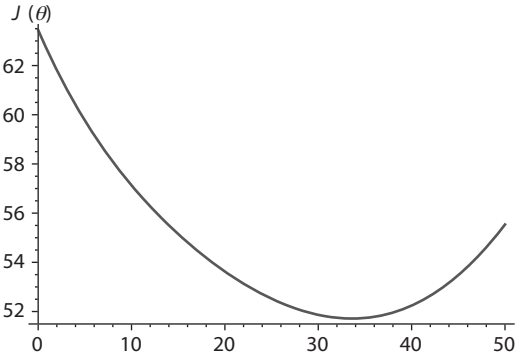


Figure 1.5. Plot of the function $J(\theta)$.

where the angles $\alpha_i(\theta)$ are as labeled in Figure 1.4. Figure 1.5 shows the plot of the time as a function of the crossing point θ.

According to Theorem 1.1, we now find the stationary points of $J(\theta)$. We use the following identities:

$$\tan\alpha_1(\theta) = \frac{\theta}{a}, \tag{1.6a}$$

$$\tan\alpha_2(\theta) = \frac{d-\theta}{b}, \tag{1.6b}$$

$$\frac{d}{d\alpha}\tan(\alpha) = \sec^2(\alpha) = \cos^{-2}(\alpha), \tag{1.6c}$$

$$\frac{d}{d\alpha}\sec(\alpha) = \sec(\alpha)\tan(\alpha). \tag{1.6d}$$

To obtain the first-order optimality condition, differentiate $J(\theta)$ and set it equal to zero:

$$J'(\theta) = \frac{a}{v_1}\,\sec(\alpha_1(\theta))\tan(\alpha_1(\theta))\,\frac{d\alpha_1(\theta)}{d\theta} + \frac{b}{v_2}\,\sec(\alpha_2(\theta))\tan(\alpha_2(\theta))\,\frac{d\alpha_2(\theta)}{d\theta} = 0.$$

By identity (1.6) it holds $\tan(\alpha_1(\theta))/\theta = 1/a = \text{constant}$. Differentiating both sides of this equation with respect to θ yields

$$\frac{1}{\theta}\sec^2(\alpha_1(\theta))\left(\frac{d\alpha_1(\theta)}{d\theta}\right) - \frac{\tan\alpha_1(\theta)}{\theta^2} = 0 \Rightarrow \frac{d\alpha_1(\theta)}{d\theta} = \frac{1}{\theta}\sin(\alpha_1(\theta))\,\cos(\alpha_1(\theta)).$$

Similarly,

$$\frac{d\alpha_2(\theta)}{d\theta} = -\frac{1}{d-\theta}\sin(\alpha_2(\theta))\,\cos(\alpha_2(\theta)).$$

Replacing these values in $J'(\theta)$ and again using the identities in (1.6), one reaches the conclusion that $J'(\theta^*) = 0$ is achieved at the unique point that satisfies

$$\frac{\sin(\alpha_1(\theta^*))}{\sin(\alpha_2(\theta^*))} = \frac{v_1}{v_2}, \tag{1.7}$$

known as Snell's Law of refraction. Going back to the person at the beach, knowing Snell's Law is not very useful because he or she still has to determine the optimal crossing point $(0, \theta^*)$, however, (1.7) gives it as an implicit solution. ⋇⋇⋇

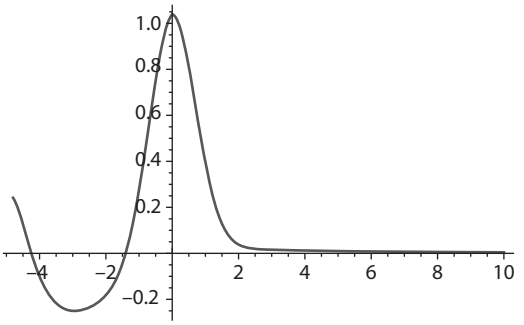


Figure 1.6. A function with "vanishing gradient."

When solving a problem of the form (1.1) analytically, one first looks for all points that satisfy (1.3). After the set of candidates is determined, one then evaluates the Hessian $\nabla^2 J(\theta)$ to verify which are local minima. If several local minima are found and if, in addition, it can be shown that $J(\theta)$ tends to ∞ as $\|\theta\|$ tends to infinity, then the location of the global minimum θ^* can be found by comparing the values of the local minima. It is worth noting that if there exists a unique stationary point, this analysis can be simplified. Indeed, if θ^* is the unique stationary point of $J(\theta)$ and if $J(\theta)$ tends to ∞ as $||\theta||$ tends to infinity, then θ^* is the unique location of the global minimum of $J(\theta)$.

Example 1.2. This example is provided to illustrate the terminology used in the field of optimization. Consider the function $J: \mathbb{R} \to \mathbb{R}$ plotted in Figure 1.6. This function has a unique global minimum that is attained at $\theta^* \approx -3$. Now consider the same function but limit its domain to $(0, \infty)$. Because $\lim_{\theta \to +\infty} J(\theta) = 0$ we find that $J(\theta)$ has 0 as an improper minimum. Indeed, 0 is not attained by any value for θ. In addition note that $J'(\theta)$ approaches 0 as θ tends to infinity. Loosely speaking we could express this by saying that "$\theta = +\infty$ is a stationary point of $J(\theta)$." The fact that $J'(\theta)$ tends to zero as θ tends to infinity is called the vanishing gradient problem, to which we shall return in later chapters. ⁂

1.2 NUMERICAL METHODS FOR UNCONSTRAINED OPTIMIZATION

In most cases, as in Example 1.1, it is impossible to solve the inversion problem $\nabla J = 0$ analytically and numerical methods are used for finding a root θ^* of $\nabla_\theta J = 0$. A numerical iterative algorithm for approximating the solution θ^* of $\nabla_\theta J = 0$ is a recursion of the form

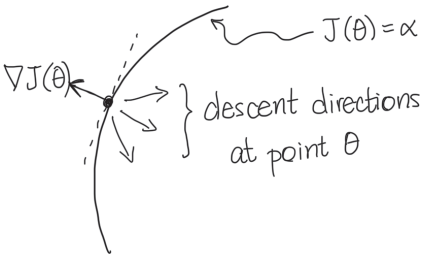
$$\theta_{n+1} = \theta_n + \epsilon_n d(\theta_n), \tag{1.8}$$

where, for each n, ϵ_n is called the *stepsize* or *gain size*, $d(\theta_n)$ is called the *direction* of the algorithm, and $\{\epsilon_n\}$ is called the *stepsize sequence* or *gain sequence*. Occasionally, ϵ_n is also referred to as *learning rate*.

Methods for approximating θ^* can be classified according to the choice of the stepsize rule and the directions. Together with an initial value θ_0 and a *stopping rule*, (1.8) constitutes a numerical algorithm that terminates hopefully close to the true optimum, where "closeness" has to be defined appropriately. Analysis of such algorithms, however, are not based on a finite termination time, but are studied as the number of iterations grows to infinity. Stopping times for the algorithm are usually based on the convergence analysis.

Definition 1.5. A *descent direction of a differentiable function* $J(\theta)$ on Θ, or *descent direction* for short, is any vector $d(\theta)$ such that $\nabla J(\theta)\, d(\theta) < 0$ for all nonstationary points $\theta \in \Theta \subseteq \mathbb{R}^d$.

A descent direction $d(\theta)$ is pointing away from the direction $\nabla J(\theta)$. Indeed $-\nabla J(\theta)\, d(\theta) > 0$ implies that there is an angle of less than 90 degrees between $d(\theta)$ and $-\nabla J(\theta)$. The figure to the right depicts the situation. Recall that $\nabla J(\theta)$ points toward the direction of growth of the function, and it can be shown that $J(\theta)$ locally decreases along any descent direction $d(\theta)$, see Exercise 1.3. For a detailed geometric interpretation of the gradient, we refer to Section A.3 in Appendix A.



Algorithms that update along a decent direction are are called *descent algorithms*. Gradient-based methods for optimization, also called *gradient descent methods*, use $d(\theta_n) = -\nabla J(\theta_n)^\top$ as a direction in the algorithm, and are a subclass of descent algorithms. There are many methods available for gradient-based optimization. Typically these algorithms are tailored to specific classes of function such as the conjugate-gradient method and variations thereof, which are suitable for optimization of quadratic functions.

For later use we state here a result showing that the negative gradient rotated by a positive definite matrix remains a descent direction.

Lemma 1.1. *Let $J(\theta)$ be a differentiable function, and let $K(\theta) \in \mathbb{R}^{d\times d}$ be a positive definite matrix for all θ, then $-K(\theta)\nabla J(\theta)^\top$ is a descent direction on $\mathbb{R}^d$.*

Proof. Since $K(\theta)$ is positive definite, it holds that $x^\top K(\theta)x > 0$ for any non-zero vector x. Letting $x^\top = \nabla J(\theta)$ shows that $\nabla J(\theta)K\nabla J(\theta)^\top > 0$. Premultiplying by -1, then yields the result. □

Newton-Raphson Method

One of the most efficient methods for unconstrained optimization is the method developed by Newton (published in 1685) and Raphson (1690). It was originally designed to find the zeroes of a polynomial. In the context of finding stationary points of $J(\theta)$, $\theta \in \mathbb{R}^d$, let the vector $G(\theta)$ represent the gradient $\nabla J(\theta)^\top$. From a point θ_n, use a linear approximation of $G(\theta)$, that is, using Taylor's expansion

$$G(\theta_{n+1}) \approx G(\theta_n) + \nabla G(\theta)(\theta_{n+1} - \theta_n),$$

where now $\nabla G(\theta) = \nabla^2 J(\theta)$ is a $d \times d$ matrix for each θ.

To approximate the zero in one step, simply set $G(\theta_{n+1}) = 0$ in the approximation and solve the right hand side for θ_{n+1}. Assuming that the inverse matrix of $\nabla G(\theta)$ exists, this yields

$$\theta_{n+1} = \theta_n - [\nabla G(\theta_n)]^{-1} G(\theta_n). \tag{1.9}$$

Comparing (1.9) with (1.8), it follows that Newton's method is a gradient descent method with adaptive stepsize sequence $\epsilon_n := \epsilon(\theta_n) = [\nabla G(\theta_n)]^{-1}$.

Theorem 1.2. *Let $J : \mathbb{R}^d \to \mathbb{R} \in C^2$ be a convex function, and assume that the Hessian is invertible. Choose an initial point $\theta_0 \in \mathbb{R}^d$ and let $\{\theta_n\}$ be the sequence defined by (1.9),*

with $G = \nabla J^\top$, *that is,*

$$\theta_{n+1} = \theta_n - [\nabla^2 J(\theta_n)]^{-1} \nabla J(\theta_n)^\top.$$

Suppose that $\bar{\theta}$ *is an accumulation point of the sequence* $\{\theta_n\}$ *such that* $\nabla^2 J(\bar{\theta}) > 0$, *then* $\bar{\theta}$ *is a local minimum of* $J(\theta)$ *and the* rate of convergence *is superlinear, that is, there exists a sequence* $\{c_n\}$ *such that* c_n *tends to zero as* n *tends to* ∞ *and for some finite* N *itholds that*

$$\|\theta_{n+1} - \bar{\theta}\| \leq c_n \|\theta_n - \bar{\theta}\|, \quad n \geq N.$$

Furthermore, if $J \in C^3$ *then the rate of convergence is quadratic, that is, there exists a constant* $c > 0$ *such that, for large* n

$$\|\theta_{n+1} - \bar{\theta}\| \leq c \|\theta_n - \bar{\theta}\|^2.$$

The proof of the result (omitted here) uses Taylor's approximation. Once the trajectory θ_n reaches a neighborhood of a local minimum θ^*, the Hessian $\nabla^2 J(\theta_n)$ becomes positive definite, which implies that it is invertible and that the Newton step moves along a descent direction, see Lemma 1.1. Although very efficient for convex functions, Newton's method has a number of practical problems when applied as a general-purpose optimization method:

- Newton's method finds zeros of the gradient, which may be locations of minima or inflection points for general functions. Consequently, it cannot be guaranteed that the Hessian is positive definite at every stationary point.
- The Hessian may not be invertible at every point.
- Finally, it needs calculation of gradients, Hessians, and Hessian inversion, all of which may be lengthy numerical operations, rendering the method slow. In some cases the Hessian can be approximately computed by repeated numerical function evaluation and we refer to Section 11.1.2 for details.

For deterministic problems, the *efficiency* of a method is defined in terms of CPU time to achieve a given precision δ. A number of algorithms have been proposed under the common name of "quasi-Newton" methods, which attempt to increase the efficiency of the method, overcoming the problems pointed out above.

⋆Cauchy's Method

The method known as *steepest descent* (or Cauchy's) for minimization of a cost function $J(\theta)$ chooses $d(\theta_n) = -\nabla_\theta J(\theta_n)$ at each iteration of (1.8). Originally proposed by Cauchy in 1847, instead of premultiplying by the matrix $[\nabla^2 J(\theta_n)]^{-1}$, the method chooses the stepsize to "move" along the direction $d(\theta_n)$ to reach the minimum on that line, that is,

$$\epsilon_n := \epsilon(\theta_n) = \arg\min_{\epsilon > 0} (J(\theta_n - \epsilon \nabla J(\theta_n)^\top)).$$

Gradient-Based Methods: Nonadaptive Stepsizes

As mentioned before, Newton's method has a good convergence rate, but every iteration may require too much computational time. Cauchy's method can have slow convergence due to possible zigzagging of the iterations, and several modifications have been proposed for adaptive stepsizes (where ϵ_n depends on $\theta_n, J(\theta_n), \nabla J(\theta_n)$, etc). Common methods use Wolfe's conditions [327, 328] and Armijo's rules [7] (and [29, 169] in combination with projection), which ensure that all accumulation points are local minima. For deterministic optimization adaptive stepsizes are undoubtedly superior to nonadaptive stepsizes. However,

the focus of the present text is to extend the basic methodology for deterministic optimization to problems where the observations of the function $J(\theta)$ and its gradients (if available) are noisy, and the noise models may be very complex. For such scenarios, non adaptive stepsizes are simpler to analyze. The gradient-based methods use $d(\theta) = -\nabla J(\theta)$ as the direction of the algorithm, and the stepsizes can be of two kinds: either decreasing: $\epsilon_n \downarrow 0$, or constant: $\epsilon_n \equiv \epsilon$.

Without any detailed analysis, inspecting the mere structure of (1.8) allows us already to deduce properties of the stepsize sequence. To see this, insert the expression for θ_n on the right-hand side of (1.8), which yields $\theta_{n+1} = \theta_{n-1} + \epsilon_n d(\theta_n) + \epsilon_{n-1} d(\theta_{n-1})$ and continuing the recurrence

$$\theta_{n+1} = \theta_0 + \sum_{i=0}^{n} \epsilon_i \, d(\theta_i).$$

Suppose that $d(\cdot)$ is bounded. Then, for the algorithm to find θ^*, the stepsizes have to satisfy

$$\sum_{n=1}^{\infty} \epsilon_n = \infty, \tag{1.10}$$

so that the sequence $\{\theta_n\}$ is not confined to some bounded set (or, equivalently, will cover any bounded set as it can potentially reach any point in $\mathbb{R}^d$). Further conditions are required in order to ensure convergence of the algorithm to the optimal θ^*, as we will show in the upcoming theorem. Before we state and prove the main result in this section, we provide a useful technical result. The result and its proof is an adaptation of Lemma 1 in [34].

Lemma 1.2. *Consider the real-valued recursion:*

$$x_{n+1} = x_n - g_n + h_n, \quad x_0 \in \mathbb{R},$$

where $g_n \geq 0$ for all n, and the sequence h_n is summable, i.e., $\sum_n |h_n| < \infty$. Then either (i) $x_n \to -\infty$ or (ii) x_n converges to a finite value and $\sum_n g_n$ converges.

Proof. Note that

$$\begin{aligned} x_{n+2} &= x_{n+1} - g_{n+1} + h_{n+1} \\ &= x_n - (g_n + g_{n+1}) + (h_n + h_{n+1}). \end{aligned}$$

Repeating this argument m times yields the telescopic sum

$$x_{m+n} = x_n - \sum_{i=n}^{m+n-1} g_i + \sum_{i=n}^{m+n-1} h_i. \tag{1.11}$$

By assumption $g_n \geq 0$, which implies

$$x_{m+n} \leq x_n + \sum_{i=n}^{m+n-1} |h_i| < \infty. \tag{1.12}$$

Use now $-\infty < \sum_{i=1}^{\infty} |h_i| < \infty$ to show that for all n

$$\limsup_{m\to\infty} \sum_{i=n}^{m+n-1} |h_i| = \lim_{m\to\infty} \sum_{i=n}^{m+n-1} |h_i| = \sum_{i=n}^{\infty} |h_i| < \infty \tag{1.13}$$

and

$$\liminf_{n\to\infty} \sum_{i=n}^{\infty} |h_i| = \lim_{n\to\infty} \sum_{i=n}^{\infty} |h_i| = 0. \tag{1.14}$$

Moreover, we have by (1.12) that

$$x_{m+n} \le x_n + \sum_{i=n}^{\infty} |h_i| < \infty. \tag{1.15}$$

By (1.13), taking the limit superior on both sides of the inequality (1.15) as m tends to ∞ yields for all n

$$\limsup_{m\to\infty} x_{m+n} \le x_n + \sum_{i=n}^{\infty} |h_i|,$$

and, since $\limsup_{m\to\infty} x_m = \limsup_{m\to\infty} x_{m+n}$, we arrive at

$$\limsup_{m} x_m \le x_n + \sum_{i=n}^{\infty} |h_i|.$$

By (1.14) together with (1.15), taking the limit inferior on both sides of the above inequality gives

$$\limsup_{m\to\infty} x_m \le \liminf_{n\to\infty} x_n < \infty,$$

and, as $\liminf_{m\to\infty} x_m \le \limsup_{m\to\infty} x_m$ by definition, we arrive at

$$\limsup_{m\to\infty} x_m = \liminf_{m\to\infty} x_m,$$

which implies that either x_n converges to some finite $\bar{x} \in \mathbb{R}$, or $x_n \to -\infty$.

In the case that $\lim_n x_n = \bar{x} \in \mathbb{R}$, letting $n = 0$ in (1.11) yields

$$\sum_{i=0}^{m-1} g_i = \sum_{i=0}^{m-1} h_i - x_m + x_0,$$

and as the right-hand side of the above equation converges as $m \to \infty$ to a finite value so does the left-hand side, which proves the claim. □

Next, we introduce two important concepts.

Definition 1.6. Let $\Theta \subset \mathbb{R}^d$ be an open connected set. A mapping $f : \Theta \to \mathbb{R}$ is called *Lipschitz continuous* if $L \in \mathbb{R}$ exists such that for any $x, x + \Delta \in \Theta$ is holds that

$$\|f(x) - f(x+\Delta)\| \le L \|\Delta\|.$$

The constant L is called *Lipschitz constant*.

Definition 1.7. We say that a sequence $\{x_n\}$ with limit $\bar{x}$ *achieves the limit in finite time* if there exist a finite index $m < \infty$ such that $x_n = \bar{x}$ for $n \ge m$.

We are now ready to state the gradient-descent theorem for decreasing stepsize.

Theorem 1.3. *Let $J \in C^2$ and assume that ∇J is Lipschitz continuous on $\mathbb{R}^d$. For given initial value θ_0, let $\{\theta_n\}$ be given through the algorithm*

$$\theta_{n+1} = \theta_n - \epsilon_n \nabla J(\theta_n)^\top, \tag{1.16}$$

where the gain sequence $\{\epsilon_n\}$, with $\epsilon_n > 0$ for all n, satisfies

$$\sum_{n=1}^{\infty} \epsilon_n = +\infty, \quad \sum_{n=1}^{\infty} \epsilon_n^2 < \infty. \tag{1.17}$$

If $\{\|\nabla J(\theta_n)\| : n \geq 0\}$ is bounded, then any (finite) limit θ^* of $\{\theta_n\}$ is a stationary point of $J(\theta)$. If, in addition, θ^* is not attained in finite time, then, for n sufficiently large, $\{J(\theta_k), k \geq n\}$ is a strictly monotone decreasing sequence.

Proof. Approximating $J(\theta_{n+1})$ via a Taylor series expansion developed at θ_n (e.g., let $\eta = \theta_{n+1} - \theta_n$ and $t = 1$ in (1.5)), yields

$$J(\theta_{n+1}) = J(\theta_n) + \nabla J(\theta_n)(\theta_{n+1} - \theta_n) + \frac{1}{2}(\theta_{n+1} - \theta_n)^\top \nabla^2 J(\xi)(\theta_{n+1} - \theta_n), \tag{1.18}$$

where $\xi = \alpha\, \theta_n + (1 - \alpha)\theta_{n+1}$ for some $\alpha \in [0, 1]$. Inserting (1.16) into the above representation of $J(\theta_{n+1})$ yields

$$J(\theta_{n+1}) = J(\theta_n) - \epsilon_n ||\nabla J(\theta_n)||^2 + \frac{\epsilon_n^2}{2} \nabla J(\theta_n) \nabla^2 J(\xi) \nabla J(\theta_n)^\top. \tag{1.19}$$

Recall that $\|\cdot\|$ denotes the Euclidean norm. Call $g_n = \epsilon_n \|\nabla J(\theta_n)\|^2$ and $h_n = \epsilon_n^2 \nabla J(\theta_n) \nabla^2 J(\xi) \nabla J(\theta_n)^\top / 2$, then

$$J(\theta_{n+1}) = J(\theta_n) - g_n + h_n.$$

From Lipschitz continuity of $\nabla J(\theta)$ it follows (see Exercise 1.7 below) that

$$|h_n| \leq \frac{\epsilon_n^2}{2} L \|\nabla J(\theta_n)\|^2,$$

for some finite constant L. Boundedness of the gradient along the trajectory together with $\sum \epsilon_n^2 < \infty$, shows that h_n is absolutely summable, so we can apply Lemma 1.2 to conclude that $J(\theta_n)$ either tends to $-\infty$, or it converges and

$$\sum_{n=0}^{\infty} \epsilon_n \|\nabla J(\theta_n)\|^2 < \infty. \tag{1.20}$$

Suppose that $\bar{\theta} \in \mathbb{R}^d$ is the limit of θ_n and achieved in finite time. Then $\theta_n = \bar{\theta}$ for all n larger than some k, which can only happen if the update $\epsilon_n \nabla J(\theta_n = \bar{\theta}) = 0$ for $n > k$. This shows that $\bar{\theta}$ is a stationary point. In case $\bar{\theta}$ is not achieved in finite time, we have from (1.20) together with continuity of ∇J that $\|\nabla J(\bar{\theta})\| = \lim_{i \to \infty} \|\nabla J(\theta_{m_i})\| = 0$. This shows that $\bar{\theta}$ is a *stationary* point.

We turn to the proof of the second part of the statement. As before, we denote the limit of θ_n by $\bar{\theta}$. We apply the bound

$$\|\nabla J(\theta_n) \nabla^2 J(\xi) \nabla J(\theta_n)^\top\| \leq L \|\nabla J(\theta_n)\|^2$$

(see Exercise 1.7) to (1.19) and thereby establish that

$$J(\theta_{i+1}) \leq J(\theta_i) - \left(\epsilon_i - \frac{1}{2} L\, \epsilon_i^2\right) \|\nabla J(\theta_i)\|^2.$$

Since ϵ_i tends to zero as i tends to infinity, we have for sufficiently large i that $L\, \epsilon_i < 2$, and thus $(\epsilon_i - L\, \epsilon_i^2/2)\|\nabla J(\theta_i)\|^2 > 0$, for $\|\nabla J(\theta_i)\| \neq 0$. Hence, if $\bar{\theta}$ is not attained in finite time, so that $\|\nabla J(\theta_i)\| \neq 0$ for all i, then there exists i_0, such that $\{J(\theta_i) : i \geq i_0\}$ is strictly monotone decreasing toward $J(\bar{\theta})$. □

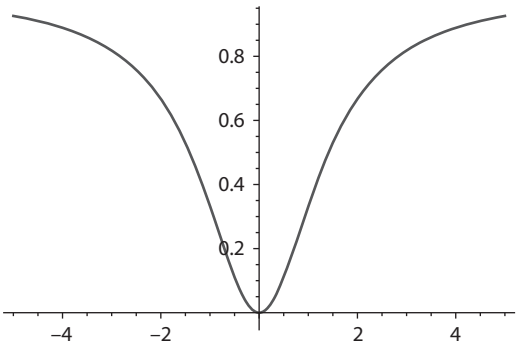


Figure 1.7. An example of a function with unique minimum and uniformly bounded gradient.

As the theorem shows, a gradient descent algorithm will find a stationary point of $J(\theta)$, but the nature of that point cannot be deduced from the algorithm alone and it requires some knowledge on the curvature of $J(\theta)$ in a neighborhood of θ^*; see Theorem 1.1. A sufficient condition for the algorithm to converge to a minimum, which then is also the unique global minimum, is convexity of $J(\theta)$. Exercise 1.12 asks to show this result.[1]

Theorem 1.3 provides sufficient conditions under which the sequence obtained via a gradient descent algorithm finds a stationary point of $J(\theta)$. Next to more generic conditions such as the choice of the stepsize and sufficient smoothness of $J(\theta)$, the key condition is that of boundedness of the gradient *along the trajectory* $\{\theta_n\}$. A nontrivial example of a mapping with bounded gradient is $J(\theta) = 1 - 2/(2+\theta^2)$, see Figure 1.7. Note that $\lim_{|\theta|\to\infty} J'(\theta) = 0$, shown in Figure 1.7.

However, typically the assumption of boundedness of the gradient along $\{\theta_n\}$ is not straightforward to check except for simple cases as the following example shows.

Example 1.3. When the gradient is not bounded for all θ, it is sometimes useful to apply the argument that the algorithm will not persistently move "away from the minimizer." For illustration, consider $J(\theta) = \theta^2 + c$, for some constant c. The minimization problem has unique solution $\theta^* = 0$ and, by computation,

$$|\theta_{n+1}| = |\theta_n - \epsilon_n J'(\theta_n)| = |\theta_n - 2\epsilon_n\theta_n| = |\theta_n(1-2\epsilon_n)| = |\theta_n|\,|1-2\epsilon_n|.$$

So, as soon as $\epsilon_n < 1/2$ for some n, we see that $|\theta_{m+1}| < |\theta_m|$ for all $m \geq n$, and the trajectory stays inside a bounded set, which implies finiteness of the gradient along the trajectory.

⋇⋇⋇

Next we discuss a more challenging example.

Example 1.4. Let us consider again the function in Example 1.2 illustrated in Figure 1.6. Following the same argument as for the function $\theta^2 + c$ in the previous example, one can show that the gradient of the function in Example 1.2 is bounded along the trajectories that start at initial values $\theta_0 < 0$. Moreover, if $\theta_0 > 0$, then the negative gradient will be

[1] While it cannot be guaranteed that the gradient descent finds a local minimum, it is worth noting that the algorithm "goes in the right direction" and the likelihood that the algorithm gets trapped at a saddle point is small in practice. See Exercise 1.15 for an example of a case where the algorithm provably gets trapped in a saddle point. The problem of convergence to a saddle point can be avoided by using specific adaptive stepsizes. However, as explained in our discussion at the beginning of this section, non adaptive stepsizes are preferable when the observations of $J(\theta)$ are noisy, which is the main focus of this monograph; deterministic optimization is introduced here, but it is not the topic of our work.

positive and θ_n becomes an increasing sequence. While it is true that the limit here satisfies $\lim_{n\to\infty} J'(\theta_n) = 0$, the corresponding limit point $\lim_{n\to\infty} \theta_n = \infty$ is not finite, and 0 is an improper minimum on $(0, \infty)$. Even worse, for initial value $\theta_0 > 0$, the descent direction moves away from the actual solution, creating a numerical instability, and the algorithm should be properly modified.

This is an important situation that arises in practical applications, because the algorithm could diverge if applied directly. Interestingly, the function in Figure 1.7 also has vanishing gradients as $\theta \to \pm\infty$, but in that case this does not pose a problem because the negative gradient is a descent direction and thus it "pulls" the sequence θ_n toward the unique minimum. ⋇⋇⋇

On occasion, it is possible to measure the outcome $J(\theta)$ of the performance of a system but the gradient $\nabla J(\cdot)$ is analytically unavailable. Instead of a gradient, some methods use a finite difference approximation. More generally, suppose that the algorithm is driven by a biased approximation of the gradient:

$$\theta_{n+1} = \theta_n - \epsilon_n\big(\nabla J(\theta_n)^\top + \beta_n(\theta_n)\big), \tag{1.21}$$

where the decreasing bias terms satisfy $\beta_n(\theta_n) \to 0$. Lemma 1.3 provides an important extension of Theorem 1.3 to biased algorithms. In the presence of bias, we have to exclude the case that for some n, the update $\nabla J(\theta_n)^\top + \beta_n(\theta_n)$ becomes zero for θ_n, so that the algorithm freezes at θ_n due to the bias.

Lemma 1.3. *Let $J \in C^2$ be such that the gradient is a Lipschitz continuous function on $\mathbb{R}^d$ and consider the biased algorithm (1.21), where the bias and the stepsize sequence $\{\epsilon_n\}$, with $\epsilon_n > 0$ for all n, satisfy*

$$\sum_{n=1}^{\infty} \epsilon_n = +\infty, \quad \sum_{n=1}^{\infty} \epsilon_n \|\beta_n(\theta_n)\| < \infty, \quad \sum_{n=1}^{\infty} \epsilon_n^2 < \infty. \tag{1.22}$$

If $\{\|\nabla J(\theta_n)\| : n \geq 0\}$ is bounded, then any (finite) limit θ^* of $\{\theta_n\}$ is a stationary point of $J(\theta)$. If, in addition, θ^* is not attained in finite time, then, for n sufficiently large, $\{J(\theta_k), k \geq n\}$ is a strictly monotone decreasing sequence.

Proof. The method of proof for this lemma is the same as for Theorem 1.3. The Taylor series for $J(\theta)$ now involves the bias terms, and under the assumptions, the corresponding terms g_n, h_n can be defined to apply Lemma 1.2. The details are left as an exercise, see Exercise 1.9. □

Lemma 1.3 can be adapted to a setting with non-vanishing bias. This is explained in the following example.

Example 1.5. Consider the biased algorithm (1.21) and assume that $\beta_n(\theta_n) = \hat{\beta}_n(\theta_n) + \beta$, for $\lim_n ||\hat{\beta}_n(\theta_n)|| = 0$ and some non-zero vector β. Hence, the bias does not asymptotically vanish. Let $\hat{J}(\theta) = J(\theta) + \beta^\top\theta$. Provided that the conditions in Lemma 1.3 are met for $J(\theta)$ and $\hat{\beta}_n(\theta_n)$, they straightforwardly extend to $\hat{J}(\theta)$ and $\hat{\beta}_n(\theta_n)$. The algorithm then finds a stationary point θ^* of $\hat{J}(\theta)$ that satisfies $\nabla J(\theta^*) + \beta^\top = 0$. Hence, the algorithm traces the stationary point of the shifted performance function $J(\theta)$. ⋇⋇⋇

Example 1.6. Suppose that we do not know the function $J(\cdot)$ analytically, but for any point θ it is possible to obtain the numerical value of $J(\theta)$. In this situation, $\nabla J(\theta)$ is not available in closed-form either. A commonly used approximation to the derivative is given by finite

differences (FD), which require that $J \in C^3$. In this example we will use a "centered" version of the approximation as follows. For simplicity, let $\theta \in \mathbb{R}$ and use a Taylor expansion around θ to obtain

$$\frac{J(\theta_n + c_n) - J(\theta_n)}{2c_n} = \frac{J'(\theta_n)}{2} + \frac{1}{4}J''(\theta_n)\, c_n + \beta_+(\theta_n, c_n)$$

$$\frac{J(\theta_n) - J(\theta_n - c_n)}{2c_n} = \frac{J'(\theta_n)}{2} - \frac{1}{4}J''(\theta_n)\, c_n + \beta_-(\theta_n, c_n)$$

so that the centered, or two-sided FD satisfies

$$\frac{J(\theta_n + c_n) - J(\theta_n - c_n)}{2c_n} = J'(\theta_n) + \beta_n(\theta_n, c_n),$$

where $\beta_n(\theta, x) = \beta_+(\theta, x) + \beta_-(\theta, x) = O(x^2)$, for fixed θ. Note that the terms containing $J''(\theta_n)$ cancel out.

When implementing FD in the descent algorithm, it is necessary to show that $\lim_{n\to\infty} \beta_n(\theta_n, c_n) = 0$ to conclude that the algorithm converges to the optimal value. Note that the main problem in showing convergence lies in the fact that we do not know beforehand the sequence $\{\theta_n\}$ visited by the algorithm. To establish convergence in (1.21), we need to verify either (a) that the third derivative $J'''(\cdot)$ is uniformly bounded in θ, or (b) that θ_n remains within a compact set along the sequence, which would imply that $J'''(\theta_n)$ is uniformly bounded (as $n \to \infty$). When either (a) or (b) hold, we know that $\beta_n(\theta_n, x) \to 0$ for any sequence $\{\theta_n\}$ visited by (1.21) as long as $x \to 0$. Hence, we can choose $c_n = O(n^{-c})$ for some constant $c > 0$, which implies $\beta_n(\theta_n, c_n) = O(n^{-2c})$. In general the choice of c_n will depend on how fast $\epsilon_n \to 0$. Assume that $\epsilon_n = O(n^{-\gamma})$, so that (1.22) holds for $\gamma \in (0, 1]$. From Lemma 1.3 it follows that Theorem 1.3 can be extended to finite difference algorithms provided that

$$\sum_{n\geq 1} \epsilon_n \beta_n < \infty \Longrightarrow \sum_{n\geq 1} n^{-(\gamma+2c)} < \infty,$$

so that we need $\gamma + 2c > 1$ for the algorithm to converge. When $\gamma = 1$, positive c is sufficient.

xxx

While gradient-based methods of the type (1.21) ensure convergence for functions with only one stationary point giving the location of the global minimum (called "unimodal") and which are continuously differentiable, the rate of convergence may be much slower than Newton's method. In particular, the steepest descent method has linear convergence, i.e., there is a constant $c \in (0, 1)$ such that $\|\theta_{n+1} - \theta^*\| \leq c\|\theta_n - \theta^*\|$, whereas Newton's method in general has quadratic convergence, see Theorem 1.2. On the other hand, the gradient descent algorithm shows remarkable resilience even for distorted gradient measurements as long as the size of the distortion decreases as $n \to \infty$, as shown in Lemma 1.3.

We complete this discussion by providing the equivalent statement to Theorem 1.3 for constant stepsize. It is worth noting that when the bias does not vanish asymptotically, the algorithm will find a stationary point of a modified objective function. Moreover, the effect a bias has on the fixed stepsize algorithm is different from the effect a bias has on the decreasing stepsize algorithm; compare Example 1.6 with the theorem below.

Theorem 1.4. *Let $J \in C^2$. Assume that $\nabla J(\theta)$ is Lipschitz continuous on $\mathbb{R}^d$ with Lipschitz constant L. Consider the constant stepsize algorithm*

$$\theta_{n+1} = \theta_n - \epsilon\, \nabla J(\theta_n)^\top,$$

for $\epsilon > 0$. Then any (finite) limit θ^* of $\{\theta_n\}$ is a stationary point of $J(\theta)$. Moreover, if (i) $\epsilon < 2/L$ and (ii) θ^* is not attained in finite time, then $\{J(\theta_n)\}$ is a strictly monotone decreasing sequence.

Let β_n denote the bias at the n-th iteration, and assume that $\lim_n \beta_n = \beta \in \mathbb{R}^d$. Then, any (finite) limit θ^*_β of $\{\theta^\beta_n\}$ given by

$$\theta^\beta_{n+1} = \theta^\beta_n - \epsilon\,(\nabla J(\theta^\beta_n)^\top + \beta_n))$$

solves $\nabla J(\theta^*_\beta) + \beta = 0$, i.e., in the asymptomatically unbiased case (given by $\beta = 0$), θ^*_β is a stationary point of $J(\theta)$, and in the asymptotically biased case (given by $\beta \neq 0$), θ^*_β is a stationary point of the adjusted objective $J(\theta) + \beta^\top \theta$.

Proof. If θ_n converges to some $\theta^* \in \mathbb{R}^d$, then this is only possible if $\lim_n \epsilon \nabla J(\theta_n) = 0$. Since ϵ is constant, this implies $\nabla J(\theta_n) = 0$, and by continuity of ∇J, is holds that $\nabla J(\theta^*) = 0$ and θ^* is thus a stationary point.

For the next part of the proof, we note that we have already shown in the proof of Theorem 1.3 that for any $i \geq 0$,

$$J(\theta_{i+1}) \leq J(\theta_i) - \epsilon \left(1 - \frac{1}{2} L\,\epsilon\right) \|\nabla J(\theta_i)\|^2.$$

Hence, for $\epsilon < 2/L$ we have that $(1 - \frac{1}{2} L\,\epsilon) > 0$ so that $J(\theta_{i+1}) < J(\theta_i)$, which shows that $J(\theta_i)$ is strictly monotone decreasing toward $J(\theta^*)$, with θ^* a stationary point.

For the biased case, we argue like before for showing that convergence of θ^β_n toward $\theta^*_\beta \in \mathbb{R}^d$ together with continuity of $\nabla J(\theta)$ implies $\nabla J(\theta^*_\beta) + \beta = 0$. This shows that θ^*_β is a stationary point of $J(\theta)$ for $\beta = 0$. For $\beta \neq 0$, note that $\nabla(J(\theta) + \beta^\top \theta) = \nabla J(\theta) + \beta^\top$, so that θ^*_β is a stationary point of $J(\theta) + \beta^\top \theta$. □

Typically, the Lipschitz constant for the gradient is hard to bound, and one applies the algorithm for ϵ "small." If in addition to the assumptions in Theorem 1.4, the function $J(\cdot)$ is convex, then the unbiased algorithm converges to the location of the minimum of J.

1.3 CONSTRAINED OPTIMIZATION

In this section we turn to optimization problems involving constraints. For ease of reference we introduce the general setting in the following definition.

Definition 1.8. For $J(\theta) \in C^1$

- the unconstrained optimization problem

$$\min J(\theta),$$

  or
- for $g_i(\theta), i = 1, \ldots, p$, and $h_j(\theta), j = 1, \ldots, q$, all in C^1, the constrained optimization problem

$$\min_{\theta \in \Theta} J(\theta), \tag{1.23}$$
$$\Theta = \{\theta \in \mathbb{R}^d : g(\theta) \leq 0, h(\theta) = 0\},$$

is called a non-linear problem (NLP). The function $J\colon \mathbb{R}^d \to \mathbb{R}$ is called the objective function, the set Θ is called the feasible region (including the case $\Theta = \mathbb{R}^d$), and a point $\theta \in \Theta$ is called a feasible point.

An NLP is called a (strictly) convex non-linear problem, or (strictly) convex problem for short, if $J(\theta)$ and—in case the problem has constraints—each $g_i(\theta), i = 1, \ldots, p$, are (strictly) convex, and each $h_j(\theta), j = 1, \ldots, q$, is an affine function (linear plus a constant).

An NLP is characterized by functions $g\colon \mathbb{R}^d \to \mathbb{R}^p$, $h\colon \mathbb{R}^d \to \mathbb{R}^q$, that represent p inequality and q equality constraints that must be satisfied. Note that since the constraints are convex by assumption, the feasible region Θ of an NLP is a convex set.

When we want to stress that a gradient or a Hessian is taken with respect to θ of a mapping with more arguments, we write ∇_θ and ∇^2_θ, respectively.

Definition 1.9. For an NLP the associated *Lagrangian* $\mathcal{L}\colon \mathbb{R}^d \times \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ is defined as

$$\mathcal{L}(\theta, \lambda, \eta) = J(\theta) + \lambda^\top g(\theta) + \eta^\top h(\theta). \tag{1.24}$$

The vectors λ and η are called *Lagrange multipliers.*

Definition 1.10. A constraint g_i of an NLP is said to be *active at a feasible point* $\theta \in \Theta$ if $g_i(\theta) = 0$. Otherwise it is said to be *inactive*. The set $A(\theta)$ of active constraints at θ contains all indices i for which $g_i(\theta) = 0$. The *constraint qualification* condition at a feasible point θ requires that the set of vectors $\{\nabla_\theta g_i(\theta), i \in A(\theta); \nabla_\theta h_j(\theta), j = 1, \ldots, q\}$ be linearly independent, and that there exist a vector $v \in \mathbb{R}^d$, $v \neq 0$, such that:

(a) $\nabla h_j(\theta) v = 0, \quad 1 \leq j \leq q,$
(b) for all $i \in A(\theta)$ it holds that $\nabla g_i(\theta) v < 0$.

Definition 1.11. A *stationary point* $(\theta^*, \lambda^*, \eta^*)$ of an NLP is a point that satisfies the *Karush Kuhn-Tucker (KKT) conditions* if

$$\nabla_\theta \mathcal{L}(\theta^*, \lambda^*, \eta^*) = 0 \tag{1.25a}$$
$$\nabla_\lambda \mathcal{L}(\theta^*, \lambda^*, \eta^*) = g(\theta^*)^\top \leq 0, \lambda^* \geq 0, \text{ and } \forall i : \lambda_i^* g_i(\theta^*) = 0 \tag{1.25b}$$
$$\nabla_\eta \mathcal{L}(\theta^*, \lambda^*, \eta^*) = h(\theta^*)^\top = 0; \tag{1.25c}$$

where $\nabla_\lambda \mathcal{L}(\theta, \lambda, \eta)$ denotes the gradient of $\mathcal{L}(\theta, \lambda, \eta)$ with respect to λ and $\nabla_\eta \mathcal{L}(\theta, \lambda, \eta)$ the gradient with respect to η. A stationary point that satisfies the KKT conditions is called a *KKT point*.

Condition (1.25b) is called the *complementary slackness* property, from this property it follows that $i \notin A(\theta)$ implies $\lambda_i = 0$. The following theorem shows that that the KKT conditions are necessary conditions for a local minimum, i.e., local minima are KKT points. Moreover, if the problem is strictly convex, then the KKT conditions are also sufficient for a global minimum. The proof is standard and a proof is omitted.

Theorem 1.5. *Assume that for a given NLP the constraint qualification holds at a local minimum θ^* of $J(\theta)$ in (1.23). Then there exist $\lambda^* \in \mathbb{R}^p, \eta^* \in \mathbb{R}^q$, such that $(\theta^*, \lambda^*, \eta^*)$ is a KKT point of the NLP. The vectors λ^* and η^* are called* Lagrange multipliers.

If, in addition, if the problem is a convex NLP, then the KKT conditions hold at θ^* if and only if θ^* is the global minimum.

Example 1.7. Many canned products in the supermarket come in cans of similar shape, where the height is the same as the diameter of the container. What is the reason for this?

Allegedly, a similar question haunted Galileo about the leather bags used by traders. Here is the answer: if a fixed volume of a given good has to be canned, the containers should be produced at minimal cost (in particular using minimal amount of material) in order to maximize your profit.

This problem can be formulated as a surface minimization problem under the fixed volume constraint. Call $\theta = (r, y)^\top$, where r is the radius and y is the height of the (cylindrical) can. Then we want to find

$$\min_{r,y} J(\theta) \stackrel{\text{def}}{=} 2(\pi r^2) + 2\pi r\, y$$
$$\text{subject to:} \qquad \pi r^2 y = V,$$

where we have expressed the total surface as the rectangular surface for the side of the can, plus the two covers. The volume V is fixed. Call $h(\theta) = \pi r^2 y - V$.

We will show how to apply Theorem 1.5 in practice. The problem fails to be convex, as neither is $J(\theta)$ convex nor is h affine, and the second part of the theorem cannot be used. Instead, we proceed as follows. First, we find the KKT points that satisfy (1.25), and then we determine which one (if several) is the global optimizer. The Lagrangian is

$$\mathcal{L}(\theta; \eta) = \mathcal{L}(r, y; \eta) = 2(\pi r^2) + 2\pi r\, y + \eta(\pi r^2 y - V).$$

Condition (1.25a) for a KKT points reads

$$\frac{\partial}{\partial r}\mathcal{L}(r, y; \eta) = 4\pi r + 2\pi y + \eta\, 2\pi r y = 0 \tag{1.26}$$

$$\frac{\partial}{\partial y}\mathcal{L}(r, y; \eta) = 2\pi r + \eta\, \pi r^2 = 0. \tag{1.27}$$

From the second equality we get $\eta^* = -2/r^*$, replacing this value in the first we get: $2r + y - 2y = 2r - y = 0$, so that $y^* = 2r^*$, which is the actual proportion found in many commercial cans.

To illustrate the mathematical method, we will finish the example. Using (1.25c), i.e., $h(\theta) = 0$, we replace $y = V/\pi r^2$ to obtain the actual solution to (1.25), namely $(r^*)^3 = V/2\pi$ and $y^* = 2r^*$. The constraint qualification holds at this (unique) KKT point. Indeed there is only one constraint, and it satisfies

$$\nabla h\,(\theta) = (2\pi r y,\ \pi r^2),$$

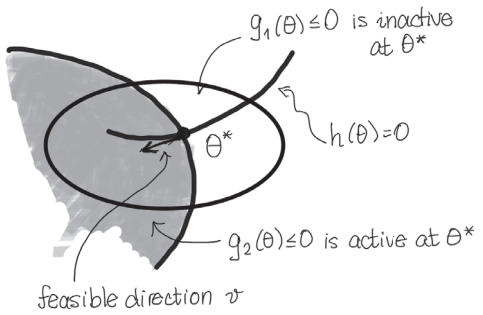
which is non-zero at r^*, y^*, as required. Observe that taking $v^\top = (r/2, -y)$ yields $\nabla h\,(\theta)\, v = 0$.

Because this is the only KKT point, it is the only candidate for the solution. To see that the KKT point is indeed a local minimum, note that $r \in (0, \infty)$ and as r either tends to 0 or to ∞, the value of $J(r, y)$ tends to ∞, so that we can conclude that J has to have a minimum for some value of $r \in (0, \infty)$. Since the only candidates for the location of a minimum are the KKT points, it follows from the uniqueness of the solution, that the KKT point is the location of the global minimum.

It is worth noting that this example is academic and placed here for illustrating the use of the theory. A more direct solution is readily obtained by direct substitution $y = V/\pi r$ into J to obtain a function of only one variable $f(r) = 2\pi(r^2 + V/\pi r^2)$. That this is convex follows from $f'(r) = 2\pi(2r - V/\pi r^2)$, and $f''(r) = 2\pi(2 + 2\,V/\pi r^3) > 0$ for all $r > 0$. The unique zero of $f'(r), r \geq 0$ is exactly at r^*. ⁂

The theorem below provides the second-order conditions that help in determining if a KKT point is indeed a local minimum along the feasible set under no convexity.

Definition 1.12. Let $(\theta^*, \lambda^*, \eta^*)$ be a stationary point of an NLP. The *critical cone* $\mathbf{C}(\theta^*, \lambda^*)$ is



$$\mathbf{C}(\theta^*, \lambda^*) = \left\{ v \in \mathbb{R}^d : \nabla g_i(\theta^*) v \le 0, \text{ if } i \in A(\theta^*), \; \lambda_i^* = 0, \nabla g_i(\theta^*) v = 0, \text{ if } \lambda_i^* > 0, \; \nabla h(\theta^*) v = 0 \right\}.$$

This cone defines the set of directions v that move along the active and equality constraints, as well as those that move "inside" the feasible set if the active constraint has a null multiplier.

Theorem 1.6. *Consider an NLP such that $J(\theta), g(\theta), h(\theta) \in C^2$ and that the constraint qualifications hold for $g(\theta), h(\theta)$ at θ^*. If $(\theta^*, \lambda^*, \eta^*)$ satisfies the first-order condition of being a stationary point (i.e., a KKT point), and the following second-order condition holds:*

$$v^\top \nabla^2_\theta \mathcal{L}(\theta^*, \lambda^*, \eta^*) v > 0, \quad \text{for } 0 \neq v \in \mathbf{C}(\theta^*, \lambda^*), \tag{1.28}$$

then θ^* is a local minimum of (1.23), where $\nabla^2_\theta \mathcal{L}$ denotes the Hessian of $\mathcal{L}$ with respect to θ.

Note that if the domain $\{\theta \in \mathbb{R}^d : g_i(\theta) \le 0, 1 \le i \le p; h_j(\theta) = 0, 1 \le j \le q\}$ is compact, then the use of the second-order condition can be avoided as continuity of $J(\theta)$ already implies existence of a global maximum and minimum on a compact set. Evaluating all stationary points then solves the optimization problem. See, for example, [49]. Lagrange multipliers frequently have an interpretation in practical contexts. In economics, they can often be interpreted in terms of prices for constraints, so-called "shadow prices" while in physics, they can represent concrete physical quantities. Mathematically, Lagrange multipliers can be viewed as *rates of change* of the optimal cost as the level of constraint changes. These types of results are called *envelop theorems* in the literature. Next, we state, without proof, the fundamental envelop theorem.

Theorem 1.7. *Consider an NLP with no inequality constraints ($p = 0$) and $J(\theta) \in C^2$ and convex. Let (θ^*, η^*) be a local minimum and Lagrange multiplier, respectively, satisfying the KKT conditions and condition (1.28). Moreoever, consider the family of continuous non-linear problems*

$$\min J(\theta), \theta \in \mathbb{R}^d \tag{1.29a}$$

$$s.t. \quad h(\theta) = u \tag{1.29b}$$

parameterized by $u \in \mathbb{R}^q$. Then there exists an open sphere S centred at $u = 0$ such that for every $u \in S$, there exist $\theta(u) \in \mathbb{R}^d$ and $\eta(u) \in \mathbb{R}^q$ such that $\theta(u)$ is the location of a local minimum of the above NLP and $\eta(u)$ the corresponding Lagrange multiplier.

Furthermore, $\theta(u), \eta(u)$ are continuously differentiable functions within S and we have $\theta(0) = \theta^*, \eta(0) = \eta^*$. In addition, for all $u \in S$,

$$\nabla_u F(u) = -\eta(u),$$

where $F(u) = J(\theta(u))$ is the optimal cost of the problem at value u.

In the case of inequality constraints, evidently $\{\theta\colon\ g(\theta) \le 0\} \subset \{\theta\colon\ g(\theta) \le u\}$ for $u > 0$. Thus, the optimal cost value of the modified problem must satisfy $F(u) \le F(0)$, for $F(u)$ defined as in Theorem 1.7. For all inactive inequality constraints, $\lambda_i = 0$, and for all active constraints, $\lambda_i > 0$, indicating a potential marginal *decrease* in the cost function as a result of increased resources.

Example 1.8. A company has a budget of \$10,000 for advertising, all of which must be spent. It costs \$3,000 per minute to advertise on television and \$1,000 per minute to advertise on radio. If the company buys x minutes of television advertising and y minutes of radio advertising, its revenue in thousands of dollars is determined by the company's data-mining oracle/statistician to be reasonably approximated by the function

$$f(x, y) = -2x^2 - y^2 + xy + 8x + 3y.$$

We can find the best solution to maximize profit solving the minimization problem:

$$\begin{aligned} \min_{x,y\in\mathbb{R}} \quad & f(x, y) = 2x^2 + y^2 - xy - 8x - 3y \\ \text{s.t.} \quad & h(x, y) = 3x + y - 10 = 0 \\ & g_1(x, y) = -x \le 0 \\ & g_2(x, y) = -y \le 0, \end{aligned}$$

where f and h are expressed in units of thousands of dollars. The Lagrangian is

$$\mathcal{L}(x, y, \lambda, \eta) = 2x^2 + y^2 - xy - 8x - 3y + \lambda_1(-x) + \lambda_2(-y) + \eta(3x + y - 10),$$

and $\nabla_{(x,y)}\mathcal{L}(x, y, \lambda, \eta) = (4x - y - 8 - \lambda_1 + 3\eta, 2y - x - 3 - \lambda_2 + \eta)^T$. By the first KKT condition, a local minimum (x^*, y^*) satisfies $\nabla_{(x,y)}\mathcal{L}(x^*, y^*, \lambda^*, \eta^*) = 0$, which gives the following simultaneous equations:

$$\begin{aligned} 4x^* - y^* - 8 - \lambda_1^* + 3\eta^* &= 0 \\ 2y^* - x^* - 3 - \lambda_2^* + \eta^* &= 0. \end{aligned}$$

There are four combinations of $g_1(x)$ and $g_2(x)$ being active/inactive.

Suppose both inequality constraints are inactive, so that complementary slackness gives $\lambda_1^* = \lambda_2^* = 0$. Together with the equality constraint, this gives three equations in three unknowns x^*, y^*, η^*. Their solutions yields the KKT point $(x^*, y^*, \lambda_1^*, \lambda_2^*, \eta^*)^\top = (\frac{69}{28}, \frac{73}{28}, 0, 0, \frac{1}{4})^\top$. This point satisfies a constraint qualification since the function $h(x, y)$ is linear, so it is a KKT point, and is thus a candidate for a local minimum. Furthermore, we have

$$\nabla^2 f(x, y) = \begin{pmatrix} 4 & -1 \\ -1 & 2 \end{pmatrix},$$

which is positive definite, and therefore is positive semi-definite (a sufficient condition for a function to be convex), thus f is convex and the KKT point is the unique global minimum

of f, and is therefore the unique global maximum of the original maximization problem. The company can therefore maximize its revenue by purchasing $\frac{69}{28}$ minutes of television time and $\frac{73}{28}$ minutes of radio time. Since we have found the unique global maximum of the optimization problem, we do not need to search for any other KKT points.

Now suppose you have in front of you this solution and the company boss puts you "on the spot" during a meeting and asks for an estimate of the extra revenue which would be generated if she spent an extra \$1,000 on advertising, what would be a reasonable answer?

Instead of solving again the problem with the budget changed to \$11,000, you can use Theorem 1.7: $-\eta$ is the instantaneous rate of change of the minimum cost function value $F(u)$ as a function of the change in the level of constraint. Here $-\eta^* = -\eta(0) = -0.25$. In terms of the original maximization problem, this translates to an *increase* of \$250 to the maximum revenue that can be generated if the the advertising budget is increased by \$1,000. Thus, knowing $\eta^* = .25$ will be enough for you to answer promptly "Madam, an extra expense of \$1,000 can only provide an extra revenue around \$250. Actually, we would be better off *decreasing* the advertising budget." ⋇⋇⋇

1.4 NUMERICAL METHODS FOR CONSTRAINED OPTIMIZATION

It should be apparent that even for seemingly small dimensions, finding all KKT points of an NLP may be an infeasible task. As in the case of unconstrained optimization, one often uses numerical iterative procedures to approximate the solution. We will now mention some of the methods that extend the simple recursive procedure (1.21). The main idea of the methods is to either approximate or reformulate the problem in terms of unconstrained optimization and then use an appropriate numerical algorithm.

Penalty Methods. These methods modify the original performance function to penalize the extent to which the constraints are not satisfied. Let $||\cdot||$ denote the Euclidean norm, then the penalized function is defined:

$$J_\alpha(\theta) = J(\theta) + \frac{\alpha}{2}\left(||g(\theta)_+||^2 + ||h(\theta)||^2\right), \tag{1.30}$$

where $g(\theta)_+ = (g_1(\theta)_+, \ldots, g_j(\theta)_+)^\top$, and $g_i(\theta)_+ = \max(0, g_i(\theta))$.

Theorem 1.8. *Consider an NLP and let* $\{\alpha_n\}$ *be an increasing sequence such that* $\lim_n \alpha_n = \infty$. *For* α_n *given, let* θ_n *be the location of the minimum of* $J_{\alpha_n}(\theta)$, *i.e., let* $\theta_n = \arg\min J_{\alpha_n}(\theta)$. *If* $\{\theta_n\}$ *has an accumulation point* θ^* *and a constraint qualification holds at* θ^*, *then* θ^* *is (feasible and) stationary for the NLP. Moreover, if* λ^*, η^* *are the Lagrange multipliers for* θ^*, *then*

$$\lambda^* = \lim_{n\to\infty} \alpha_n\,(g(\theta_n))_+, \quad \eta^* = \lim_{n\to\infty} \alpha_n\, h(\theta_n),$$

and for each $i = 1, \ldots, j$, $\lambda_i^* \geq 0$ *and* $\lambda_i^* = 0$ *if* $g_i(\theta^*) < 0$.

Figure 1.4 illustrates the idea of penalizing the unsatisfaction of the constraint. Here $J(\theta) = \theta^2$, and the constraint is $\theta \geq 3$. Naturally for this example direct inspection yields that (a) the constraint must be active at the optimal value (because the unconstrained optimum is infeasible), and (b) thus $\theta^* = 3$. The function $J_\alpha(\theta)$ looks like $J(\theta)$, except that the segment of the curve to the left of $\theta = 3$ (in the infeasible region) is "lifted" more dramatically as α increases.

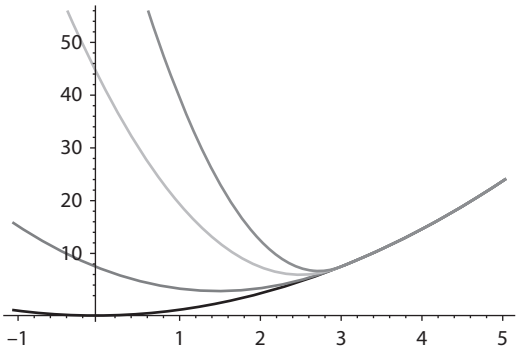


Figure 1.8. Functions $J_\alpha(\theta)$, $\alpha = 0, 1, 5, 10$.

The actual optimal values for the consecutive penalties are $0.0, 1.4, 2.5$, and 2.8. It follows from Theorem 1.8 that as $\alpha \to \infty$ this sequence converges to the optimal value. Theorem 1.8 requires solving each unconstrained problem $\min J_{\alpha_n}(\theta)$ exactly. However, this is often not possible, so one may use a gradient–based iterative method, for example, in order to *approximate* the solution θ_n. This is often referred to as "inexact optimization" for each step.

Numerical methods with inexact optimization typically use $\theta^n_{k+1} = \theta^n_k - \epsilon_k \nabla J_{\alpha_n}(\theta^n_k)^\top$, with $k = 1, 2, \ldots, T_n$, for minimizing $J_{\alpha_n}(\theta)$ with respect to θ. The terminal time T_n is either chosen to satisfy a stopping criterion, or sometimes an increasing sequence $T_n \to \infty$ is used. The idea is to approach the true solutions for the subsidiary problems as n increases, while using fewer iterations at first. The algorithm is

$$\theta^n_{k+1} = \theta^n_k - \epsilon_k \nabla J_{\alpha_n}(\theta^n_k)^\top, k = 0, \ldots T_n - 1 \tag{1.31a}$$

$$\theta^{n+1}_0 = \theta^n_{T_n} \tag{1.31b}$$

$$\alpha_{n+1} = \alpha_n + \delta_n, \tag{1.31c}$$

where $\sum \delta_n = +\infty$, and

$$\nabla J_{\alpha_n}(\theta_n) = \nabla_\theta J(\theta_n) + \alpha_n \big(g(\theta_n)^\top \nabla g(\theta_n) \mathbf{1}_{\{\|g(\theta_n)\|>0\}} + h(\theta_n)^\top \nabla h(\theta_n) \big). \tag{1.32}$$

Under appropriate conditions, the sequence $\theta^n_{T_n}$ will converge to the constrained optimum. Different stopping schemes yield different overall rates of convergence. Notice that setting the initial value for step $n+1$ as the final value for the previous step is more convenient than re-initializing, provided that the final estimate $\theta^n_{T_n}$ is indeed close to the exact optimal value for J_{α_n}. Algorithm 1.1 corresponds to the updating scheme in (1.31).

When T_n is increasing, convergence of the auxiliary optimization problem for α_n ensures that the end point $\theta^n_{T_n}$ gets closer to the minimum of J_{α_n}; however, it also implies longer running times for Algorithm 1.1 than using a constant value. Indeed, the running time for this algorithm is proportional to $\sum_{n=1}^{\tau} T_n$, considering that each iteration inside the **for** loop has constant running time. Here, τ represents the stopping time, which is usually dependent on the current values of the gradients and consecutive end points $\theta^n_{T_n}$. Amortized analysis yields that the running time per (outer) iteration is the average of the consecutive lengths T_n, which grows as $n \to \infty$ unless T_n are constant. This may produce slow algorithms.

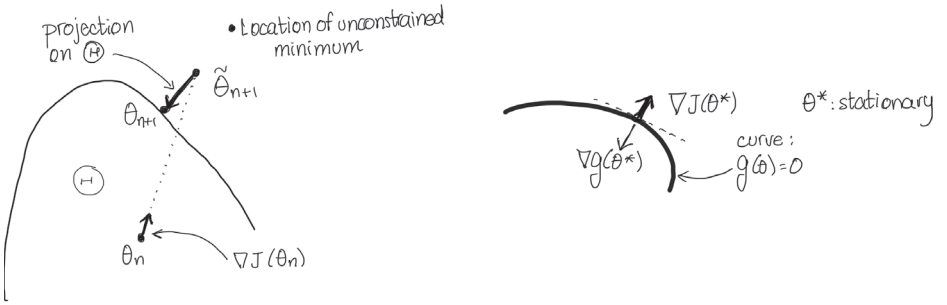


Figure 1.9. Visualization of the projection algorithm.

Algorithm 1.1 Penalty method

Read cost and constraint functions J, g, h.
Pre-define the increasing function $\textsc{Alpha}(n)$.
Pre-define the non-decreasing sequence T_n.
Pre-define the function $\textsc{Grad}(\alpha, \theta)$ that returns $\nabla J_\alpha^\top(\theta)$ in (1.32)
Initialize $\theta[0,0]$, $\alpha_0 = \textsc{Alpha}(0)$, $n = 0$
while (**not** stopping-condition) **do**
  for $(k = 0, \ldots, T_n - 1)$ **do**
    $\theta[n, k+1] = \theta[n,k] - \epsilon_k \, \textsc{Grad}(\alpha_n, \theta[n,k])$
  $\theta[n+1, 0] = \theta[n, T_n]$
  $\alpha_{n+1} = \textsc{Alpha}(n+1)$
  $n \leftarrow n+1$

Alternatively, one can introduce a two-timescale method. Let $T_n = 1$ and suppose that α_n grows ever so slowly that it "looks" constant for the iteration in θ_n when using Taylor expansions. The corresponding algorithm is of the form

$$\theta_{n+1} = \theta_n - \epsilon_n \nabla J_{\alpha_n}(\theta_n)^\top \tag{1.33a}$$

$$\alpha_{n+1} = \alpha_n + \delta_n, \tag{1.33b}$$

with $\delta_n \epsilon_n \to 0$, $\sum \delta_n = +\infty$, and $\nabla J_{\alpha_n}(\theta_n)$ as in (1.32). A convergence proof for this scheme with fixed ϵ_n, δ_n is provided in Theorem 2.12. For a treatment of general convergence results for two-timescale algorithms we refer to [222]. Algorithm 1.2 shows the pseudocode for the two-timescale implementation of the penalty method. In terms of the running time, it is now linear in the number of iterations performed in the **while** loop. However convergence of the two-timescale in terms of the stopping time τ may be slower than Algorithm 1.1 because α_n grows now very slowly. Inspecting Figure 1.4 it becomes apparent why the growth-rate of α_n has to be chosen with care: if α_n is too large, the penalty may push the algorithm far to the right (and away from the solution) and thus renders the method numerically inefficient.

Projection Methods. Gradient projection methods iterate successive solutions in the direction of improvement of the cost function (descent directions) where the value at each iteration *remains always feasible*. The projection method was introduced in [130, 131] and independently thereof in [202]. In the literature, the projection method also goes under the name *Goldstein-Levitin-Polyak projection method*, see [29, 141]. The algorithm is in

Algorithm 1.2 Penalty method: Two-timescale

Read cost and constraint functions J, g, h.
Pre-define the decreasing function $\textsc{Delta}(n)$.
Pre-define the non-decreasing sequence T_n.
Pre-define the function $\textsc{Grad}(\alpha, \theta)$ that returns $\nabla J_\alpha^\top(\theta)$ in (1.32)
Initialize $\theta_0, \alpha_0 = \textsc{Alpha}(0), n = 0$
while (**not** stopping-condition) **do**
  $\theta_{n+1} = \theta_n - \epsilon_n \, \textsc{Grad}(\alpha_n, \theta[n, k])$
  $\alpha_n = \alpha_n + \textsc{Delta}(n)$
  $n \leftarrow n + 1$

general form:

$$\tilde{\theta}_{n+1} = \theta_n - \epsilon_n \nabla J(\theta_n)^\top \tag{1.34a}$$

$$\theta_{n+1} = \Pi_\Theta\Big(\tilde{\theta}_{n+1}\Big), \tag{1.34b}$$

where $\Pi_\Theta(v)$ is the projection of the vector $v \in \mathbb{R}^d$ onto the set Θ; and for $\Theta \subset \mathbb{R}^d$ a closed convex set, the projection Π_Θ on Θ is defined as

$$\Pi_\Theta(x) = \arg\min_{z \in \Theta} ||x - z||. \tag{1.35}$$

In words, $\Pi_\Theta(v)$ is the point closest to v in Θ in Euclidean distance. Figure 1.9 (left) shows the geometric interpretation of the algorithm.

For mathematical analysis of the projection algorithm the following representation of the projection version of the gradient descent algorithm will be used in later chapters

$$\theta_{n+1} = \theta_n - \epsilon_n \big(\nabla J(\theta_n) + Z(\epsilon_n, \theta_n, -\nabla J(\theta_n))\big), \tag{1.36}$$

where where $Z(\epsilon_n, \theta_n, -\nabla J(\theta_n))$ is the "projection force" that keeps the algorithm on Θ. Specifically, if $\tilde{\theta}_{n+1} \in \Theta$, then

$$Z(\epsilon_n, \theta_n, -\nabla J(\theta_n)) = 0,$$

and otherwise

$$Z(\epsilon_n, \theta_n, -\nabla J(\theta_n)) = \frac{1}{\epsilon_n}\left(\theta_n - \epsilon_n \nabla J(\theta_n) - \Pi_\Theta\Big(\theta_n - \epsilon_n \nabla J(\theta_n)\Big)\right). \tag{1.37}$$

Note that by (1.35) the projection force on a convex set at some point $\theta \in \Theta$ is by construction no larger than the unconstrained increment given by the gradient at θ times the gain size, that is,

$$\|Z(\eta, \theta, -\nabla J(\theta))\| \le \eta \, ||\nabla J(\theta)||, \tag{1.38}$$

for $\eta > 0$, and that the projection force is monotone decreasing in the gain size

$$\|Z(\eta, \theta, -\nabla J(\theta))\| \le \|Z(\hat{\eta}, \theta, -\nabla J(\theta))\|, \tag{1.39}$$

for $\eta \le \hat{\eta}$, which stems from the fact that the force pushing the update outside of Θ is the negative gradient scaled by the gain size, and is therefore monotone decreasing in the gain size.

The actual evaluation of the projection operation is usually the main computational burden for each step in the algorithm. See, for example, [77], where the projection onto a

simplex is provided. The simplest case is that of projection on a hypercube or a hyperball, which are detailed in the following examples.

Example 1.9. In case Θ is a d-dimensional hypercube, i.e., $\Theta = [-M, M]^d$ for some finite M, the projection is easily obtained through

$$\Pi_M(\theta) := \Pi_{[-M,M]^d}(\theta) = \left(\max(\theta_i, -M)\mathbf{1}_{\{\theta_i \le 0\}} + \min(\theta_i, M)\mathbf{1}_{\{\theta_i \ge 0\}} : 1 \le i \le d \right)^\top.$$

In the special case of the projection on a hypercube we call the projection a *truncation* (on each coordinate). We call the constraint set Θ *box constraints*. Note that Θ can be encoded in the KKT setting, see (1.23), through $g_i(\theta) = \theta_i - M$ and $g_{i+d}(\theta) = -\theta_i - M$, for $1 \le i \le d$. ⁂

Example 1.10. In case Θ is a d-dimensional ball around the origin of radius $r > 0$, i.e.,

$$\Theta = B_r := \{\theta \in \mathbb{R}^d : ||\theta|| \le r\}, \tag{1.40}$$

the projection is obtained by rescaling vectors ourside of B_r:

$$\Pi_r(\theta) := \Pi_{B_r}(\theta) = \begin{cases} r\theta/||\theta|| & \text{if } \theta \notin B_r, \\ \theta & \text{if } \theta \in B_r. \end{cases} \tag{1.41}$$

Note that B_r can be encoded in KKT setting, see (1.23), through $g(\theta) = ||\theta|| - r$. ⁂

In the following we consider NLP's without equality constraints, in which case the NLP becomes

$$\min_{\theta \in \Theta} J(\theta), \qquad \Theta = \{\theta \in \mathbb{R}^d : g(\theta) \le 0\} \tag{1.42}$$

and solutions are characterized by the KKT conditions.

Before stating (a version) of the convergence result, we will motivate the result by the following consideration. Suppose that $\{\theta_n\}$ is obtained via (1.34), then the following cases can occur: (i) the minimizer θ^* is an inner point of Θ and the algorithm will (after possibly finitely many projections) stay inside Θ, and will behave just like the unconstrained version; (ii) the *unconstrained* minimizer $\tilde{\theta}^*$ lies outside of Θ (or on the boundary of Θ) and the algorithm will eventually converge to a point θ^* on the boundary of Θ; and finally (iii) the problem may be ill-posed so that θ_n has no accumulation points at all (e.g., minimizing $J(\theta) = -\theta^2$). Note that case (iii) is ruled out if we assume Θ to be compact which is a consequence of the Weierstrass theorem, see Theorem A.2 in Appendix A. Before turning to the study of the behavior of the algorithm, we provide some details on case (ii). For ease of argument we consider the fixed ϵ version of the algorithm. Suppose that θ^* lies outside Θ, and suppose that the algorithm converges to a point θ', then $\nabla J(\theta_n) + Z(\epsilon, \theta_n, -\nabla J(\theta_n))$ tends to zero as n tends to infinity. Assume, for simplicity, that only one constraint g_i is active at θ', i.e., $g(\theta') := g_i(\theta') = 0$. Then, the descent direction in θ' is $-\nabla J(\theta')^\top$. This implies that $-\nabla J(\theta')^\top$ is pointing outward of Θ. For the algorithm to have θ' as fixed point, it must hold that the projection of $\tilde{\theta}' = \theta' - \epsilon \nabla J(\theta')^\top$ on Θ is θ' itself. This means that $\tilde{\theta}' - \theta'$ is perpendicular to the tangent plane (an object in $\mathbb{R}^{d+1}$) to Θ at θ'. Since Θ is given as $\{\theta \in \mathbb{R}^d : g(\theta) \le 0\}$, we know that $\nabla g(\theta)$ is a the projection of the normal vector to the tangent plane onto the parameter space $\mathbb{R}^d$, and due to the inequality we have that $\nabla g(\theta)$ is pointing outward of Θ. This shows that $-\nabla J(\theta')^\top$ and $\nabla g(\theta')$ are co-linear and pointing in the same direction. Hence, $-\nabla J(\theta')^\top = \lambda \nabla g(\theta')$, for some $\lambda > 0$, as illustrated in Figure 1.9. We conclude that θ' is a KKT point. Note that it thus holds that $Z(\epsilon, \theta', -\nabla J(\theta'))$

and $\nabla g(\theta')$ are co-linear for all ϵ; however, they point in opposite directions. To summarize, for $\nabla J(\theta') \neq 0$ and $g(\theta') = 0$, we have $\lambda \nabla g(\theta') = -\nabla J(\theta')$, which shows that θ' is a KKT point for (1.42) and under appropriate smoothness conditions a local minimizer for (1.42).

In the presence of bias, it may happen that the biased version is co-linear with the projection force at some point θ_n so that the algorithm does not advance any more (i.e., $\theta_{n+m} = \theta_n$ for $m \geq 1$) while the gradient is not co-linear with the projection force and θ_n is thus not a KKT point. As illustrating example for this phenomena consider the coordinate descent gradient

$$G(\theta) = \mathbf{e}_j (\partial J(\theta)/\partial \theta_j),$$

where $\mathbf{e}_j$ is the jth unit vector and

$$j = \arg\max_i |\partial J(\theta)/\partial \theta_i|.$$

It is easily seen that $-G(\theta)$ is a descent direction and, in general, a biased version of $-\nabla J(\theta)$. Let Θ be a hypercube. Suppose that θ_n is the first time that the algorithm steps outside hyercube Θ, so that $\tilde{\theta}_n$ is on the surface of the hypercube. Since $G(\theta)$ is by construction perpendicular to the surface of the hypercube, this implies that algorithm gets stuck at $\tilde{\theta}_n$, i.e., $\tilde{\theta}_n = \tilde{\theta}_{n+k}$ for $k \geq 0$. Letting k now tend to ∞, neither the value of $\tilde{\theta}_{n+k}$ nor that of the bias will change. Hence, the bias cannot tend to zero and we can rule this out by imposing the condition that β_n tends to zero as n tends to ∞. If, on the other hand, the algorithm comes to a halt at $\tilde{\theta}_n$ with $\beta_n = 0$, we have found a KKT point.

We now present the theorem.

Theorem 1.9. *Consider the NLP*

$$\min_{\theta \in \Theta} J(\theta), \qquad \Theta = \{\theta \in \mathbb{R}^d : \, g(\theta) \leq 0\} \tag{1.43}$$

with Θ being a compact and convex set, and let $J(\theta) \in C^2$ with L denoting the uniform Lipschtiz constant of ∇J on Θ. Consider the algorithm

$$\tilde{\theta}_{n+1} = \theta_n - \epsilon_n (\nabla J(\theta_n)^\top + \beta_n)$$
$$\theta_{n+1} = \Pi_\Theta \left(\tilde{\theta}_{n+1} \right),$$

with either

$$\sum_n \epsilon_n = \infty, \sum_{n=1}^{\infty} \epsilon_n \, \|\beta_n(\theta_n)\| < \infty \quad \textit{and} \quad \sum_n \epsilon_n^2 < \infty,$$

where $\epsilon_n > 0$ for all n, or

$$0 < \epsilon_n = \epsilon < 2/L, \textit{ for } n \geq 0, \textit{ and } \lim_{n \to \infty} ||\beta_n|| = 0.$$

Then every accumulation point of $\{\theta_n\}$ of this algorithm is a KKT-point of the NLP in (1.43).

Proof. We proof the theorem in case of no bias. The extension to the biased case follows the line of argument provided in the discussion prior to the theorem.

As Θ is compact, then by the Bolzano-Weierstrass theorem, $\{\theta_n\}$ has accumulation points. Let θ^* be an accumulation point of $\{\theta_n\}$ and assume that θ^* is an inner point of Θ. Let $\theta_m := \theta_{n_m}$ denote the subsequence converging toward θ^*. Then, for N sufficiently large, $\theta_m \in \hat{\Theta}$, for $m \geq N$, for some compact proper subset $\hat{\Theta}$ of Θ (i.e., $\hat{\Theta}$ contains no boundary

points of Θ). Continuity of the gradient and the Hessian implies that the gradient as well as the Hessian are bounded on $\hat{\Theta}$. We now apply the arguments put forward in the proof of Theorem 1.3 for the decreasing ϵ case and Theorem 1.4 for the fixed ϵ case, to show that

$$\lim_{m\to\infty} \nabla J(\theta_m) = 0 = \nabla J(\theta^*),$$

which shows that θ^* is a stationary point of $J(\theta)$. We have assumed that θ^* is an inner point of Θ, so that $g_i(\theta^*) < 0$ for all i, and it follows that θ^* is a KKT point for (1.42).

Now consider the case that θ^* lies on the boundary of Θ. Convergence of θ_m implies that $\|\nabla J(\theta_m) + Z(\epsilon, \theta_m, -\nabla J(\theta_m))\|$ converges toward zero. Note that projection on a convex set is continuous; see Exercise 1.5. By continuity of both gradient and projection it holds that

$$\lim_{m\to\infty} \|\nabla J(\theta_m) + Z(\epsilon, \theta_m, -\nabla J(\theta_m))\| = \|\nabla J(\theta^*) + Z(\epsilon, \theta^*, -\nabla J(\theta^*))\| = 0. \tag{1.44}$$

For ϵ sufficiently small, we apply Theorem 1.4 to conclude from the above that either $\nabla J(\theta^*) = 0$ (and therefore $Z(\epsilon, \theta^*, -\nabla J(\theta^*)) = 0$) and $g_i(\theta^*) = 0$, or $\nabla J(\theta^*) \neq 0$ in which case the negative gradient points outward from Θ. For the projection force to counter balance $-\nabla J(\theta^*)$, the negative gradient has to be perpendicular to the projection of hyperplane spanned by any g_i at θ^* onto $\mathbb{R}^d$. As, moreover the $\nabla g_i(\theta^*)$'s are pointing outward of Θ, we have that $-\nabla J(\theta^*) = \sum \lambda_i \nabla g_i(\theta^*)$ for some constants $\lambda_i > 0$ where the sum runs through the indices of the active constraints. This shows that θ^* is a KKT point for (1.42).

For the decreasing ϵ we take N such that $\epsilon_n \leq 2/L$ for $n \geq N$, and we use $\|Z(\epsilon_n, \theta_n, -\nabla J(\theta_n))\| \leq \|Z(\epsilon, \theta_n, -\nabla J(\theta_n))\|$ for $n \geq N$, which stems from the fact that the projection force is monotone in the gain size; see (1.39). The proof then follows from (1.44).

To conclude the proof, we evoke Theorem 1.5, to show that any KKT point for (1.42) is the location of a local minimum for the NLP in (1.42). □

Remark 1.2. In case that Θ represents hard constraints so that $J(\theta)$ is not defined outside of Θ, the gradient of $J(\theta)$ is only defined on interior points of Θ. As on the boundary of Θ only the directional derivatives along directions pointing inward of Θ are defined (and not the gradient as such), the projection method as presented here cannot be straightforwardly applied.

Remark 1.3. Unless the constraint set Θ is of a particular nice and simple form (e.g., a "box" or a "ball"), the projection step may require numerical approximation methods (see Example 1.11 below). In general, the projection method provides an analytically attractive tool that we will use extensively in the rest of this book. Indeed, many technical assumptions become less restrictive if the algorithm is projected onto a bounded set. For example, the condition that the gradient is bounded along trajectories in Theorem 1.3 rules out even a quadratic form of $J(\theta)$, but there is nothing wrong with a quadratic function as long as the trajectories remain inside a bounded set. In many cases, we study the projected version of the algorithm restricting the solutions $\{\theta_n\}$ to a hypothetical large hyperball; see Example 1.10. If the projected algorithm has accumulation points that are independent of the hyperball, then theoretical arguments can be used to establish that the original (unprojected) version has the same limiting behavior. It is worth mentioning that the projection method is well-studied in the area of deterministic optimization. A method for finding a projection on a general convex set through iterative projection on simpler convex sets is Dykstra's method [43], and for an exhaustive overview of these kind of methods we refer to [67]. It is worth noting that projection can

be avoided by moving only along an update direction that stays inside the feasible set. An example of an algorithm that elaborates on this idea and that is popular in machine learning is the Frank-Wolfe algorithm, which uses a linear approximation of the objective for finding a descent direction that stays inside the feasible set; see [99].

We conclude this section on the projection method with a discussion on finding (approximate) projections when the feasible set is not of a simple form and the projection operation cannot be expressed analytically in closed form.

Example 1.11. In the case that the constraint function g is affine, Example 4.3 in Chapter 3 of [31] shows that the dual of this problem leads to a much simpler optimization problem with only positivity constraints. We now refer to [190] where a method is proposed for general $g \in C^2$ using the fact that $\theta_n \in \Theta$, and $\tilde{\theta}_{n+1} - \theta_n$ is of order ϵ_n so a Taylor approximation can be used to linearize the constraints around θ_n:

$$g(x) \approx g(\theta_n) + \nabla g(\theta_n)(x - \theta_n).$$

Let $v = \tilde{\theta}_{n+1}$. The constraint $x \in \Theta$ is approximated by the constraint

$$g(\theta_n) + \nabla g(\theta_n)x \leq \nabla g(\theta_n)\theta_n.$$

Call $\mathbb{A} = \nabla g(\theta_n)^\top$, and $b = \mathbb{A}\theta_n - g(\theta_n)$. The approximated, or "surrogate" subsidiary problem becomes

$$\min_x \left(\frac{1}{2} x^\top x - v^\top x \right) \tag{1.45}$$

$$\text{s.t.} \quad \mathbb{A}x \leq b, \tag{1.46}$$

The Lagrangian for this subsidiary problem is

$$\mathcal{L}(x, \mu) = \frac{1}{2} x^\top x - v^\top x + \mu^\top \mathbb{A}x - \mu^\top b.$$

Using Lagrange duality (Theorem 1.11), we seek $\max_{\mu \geq 0} (\min_{x \in \mathbb{R}^d} \mathcal{L}(x, \mu))$. Because of the quadratic form, we can solve the minimization step analytically by setting the gradient to zero, which readily yields $x^*(\mu) = v - \mathbb{A}^\top \mu$. Then the subsidiary problem is

$$\max_{\mu \geq 0} \left(\frac{1}{2} (v - \mathbb{A}^\top \mu)^\top (v - \mathbb{A}^\top \mu) - b^\top \mu \right),$$

which, after replacing the appropriate values, gives another quadratic maximization problem with a projection to the positive real numbers. Finding the zero of the derivative of this function, however, now requires an inversion of matrices depending on $g(\theta_n)$ and $\nabla g(\theta_n)$, which is generally computationally expensive. Instead, [190] propose a recursive gradient method to solve for μ. With this, one sets $\theta_{n+1} = v - \mathbb{A}^\top \mu^* = \tilde{\theta}_{n+1} - \nabla g(\theta_n)^\top \mu^*$.

The above analysis requires exact optimization for the subsidiary problem (i.e., solving μ^*), but this is often not possible, so inexact optimization can be used. Let T_n be the number of iterations used to approximate μ^* at step n of the procedure. Then the algorithm is

$$\tilde{\theta}_{n+1} = \theta_n - \epsilon_n \nabla J(\theta_n)^\top \tag{1.47a}$$

$$\mu^n_{k+1} = \max\left(0, \mu^n_k + \delta_k \left(\nabla g(\theta_n) \nabla g(\theta_n)^\top \mu^n_k + \nabla g(\theta_n)(\tilde{\theta}_{n+1} - \theta_n) - g(\theta_n)\right)\right); \quad k = 0, \ldots, T_n - 1 \tag{1.47b}$$

$$\theta_{n+1} = \tilde{\theta}_{n+1} - \nabla g(\theta_n)^\top \mu^n_{T_n}. \tag{1.47c}$$

Indeed, recalling our definitions $\mathbb{A} = \nabla g(\theta_n)^\top$, $v = \tilde{\theta}_{n+1}$, and $b = \mathbb{A}\theta_n - g(\theta_n)$ one notes that the second iteration above refers to the approximate solution of the subsidiary problem.

Because of the approximation of the non-linear constraint, this new point may be infeasible, but under appropriate conditions on $T_n, \epsilon_n, \delta_n$, this algorithm will converge. It is also possible to adapt this algorithm to a two-timescale version. ⋇⋇⋇

Multiplier Methods. These methods are based on the following result for equality constraint problems; for a proof, we refer to [31].

Theorem 1.10. *Consider an equality constrained problem. Let*

$$\theta_n^* = \arg\min_\theta \mathcal{L}(\theta, \eta_n)$$
$$\eta_{n+1} = \eta_n + \rho_n h(\theta_n^*),$$

for a sequence $\rho_n \to \infty$, then $(\theta_n^*, \eta_n) \to (\theta^*, \eta^*)$ a local minimum and a KKT point of (1.23).

The *inexact* multipliers methods use an approximation to θ_n^* via Theorem 1.3. Let T_n be a stopping time for the approximation at step n. Then the algorithm is

$$\theta_{k+1}^n = \theta_k^n - \epsilon_k \nabla_\theta \mathcal{L}(\theta_k^n, \eta_n)^\top = \theta_k^n - \epsilon_k \left(\nabla_\theta J(\theta_k^n)^\top + \nabla_\theta h(\theta_k^n)\eta_n\right); \quad k = 0, \ldots, T_n \tag{1.48a}$$
$$\eta_{n+1} = \eta_n + \rho_n h(\theta_{T_n}^n). \tag{1.48b}$$

This method can be applied to inequality constraints as well via a transformation; see [31]. The choice of T_n will determine the convergence properties. It is common to either use a stopping criterion in terms of $\nabla\mathcal{L} \approx 0$, or to use an increasing sequence T_n. Compared with the analysis of the penalty method, one can also deduce that the amortized running time will be an average of the batch lengths T_n, which is an increasing function of n.

Algorithm 1.3 Multiplier method

Read cost and constraint functions J, h.
Pre-define the increasing function Rho(n).
Pre-define the non-decreasing sequence T_n.
Initialize $\theta[0,0], \eta_n, \rho_0$ =Rho$(0), n = 0$
while (**not** stopping-condition) **do**
  for $(k = 0, \ldots, T_n - 1)$ **do**
    $\theta[n, k+1] = \theta[n,k] - \epsilon_k \left(\nabla J(\theta)^\top (\theta[n,k] + \nabla h(\theta[n,k])\, \eta_n\right)$
  $\theta[n+1, 0] = \theta[n, T_n]$
  $\eta_{n+1} = \eta_n$+ Rho$(n)\ h(\theta[n+1, 0])$
  $n \leftarrow n+1$

A two-timescale algorithm can be implemented here, like for the penalty method, using $T_n = 1$ but making ρ_n grow "slower" than ϵ_n decreases so that the primal variable behaves locally in bounded intervals as if it was driven with a constant dual variable. This algorithm will have a constant amortized running time.

Lagrange Duality Methods. In both penalty and multiplier methods, the theory establishes convergence only when an exact minimization takes place for given multiplier values. The

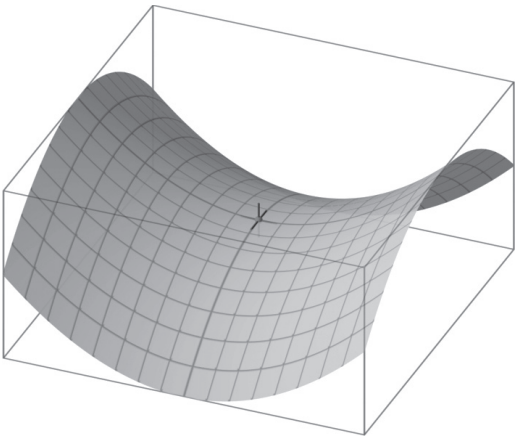


Figure 1.10. Saddle point illustration for θ, λ on the axes, no equality constraints.

numerical approximations often use inexact minimization by updating the decision variable θ_n for T_n iterations and then updating the multipliers. However, there is no guarantee that the algorithm will converge, and it is not clear how to tune the parameter T_n for better convergence.

An important class of methods is based on Lagrange Duality Theory. It is straightforward to note that the solution to (1.23) is the same as the solution of the minmax problem:

$$\min_{\theta\in\mathbb{R}^d}\max_{\lambda\geq 0,\eta}\mathcal{L}(\theta;\lambda,\eta)=\min_{\theta\in\mathbb{R}^d}\begin{cases}J(\theta) & \text{if } g(\theta)\leq 0, h(\theta)=0,\\ +\infty & \text{otherwise.}\end{cases}$$

However, the above minmax problem is clearly not useful for an iterative algorithm. Instead, we use the following strong result.

Theorem 1.11 (Saddle Point Theorem). *For a given convex NLP, the triplet* $(\theta^*,\lambda^*,\eta^*)$ *is a KKT point if and only if it is a saddle point of the Lagrangian, that is,*

$$\mathcal{L}(\theta^*,\lambda,\eta)\leq\mathcal{L}(\theta^*,\lambda^*,\eta^*)\leq\mathcal{L}(\theta,\lambda^*,\eta^*)$$

for every $\theta\in\mathbb{R}^d, \lambda(\geq 0)\in\mathbb{R}^p, \eta\in\mathbb{R}^q$. *Furthermore,*

$$\min_{\theta\in\mathbb{R}^d}\max_{\lambda\geq 0,\eta}\mathcal{L}(\theta;\lambda,\eta)=\max_{\lambda\geq 0,\eta}\min_{\theta\in\mathbb{R}^d}\mathcal{L}(\theta;\lambda,\eta).$$

The saddle point theorem can be used to maximize first over the multipliers, and then perform a minimization over the decision variables. This is the motivation for the *Uzawa* algorithm [304]:

$$\theta_{n+1}=\arg\min_{\theta}\mathcal{L}(\theta,\lambda_n,\eta_n) \tag{1.49a}$$

$$\lambda_{n+1}=\lambda_n+\max(0,\lambda_n+\epsilon_n\nabla_\lambda\mathcal{L}(\theta_{n+1},\lambda_n,\eta_n)^\top) \tag{1.49b}$$

$$\eta_{n+1}=\eta_n+\epsilon_n\nabla_\eta\mathcal{L}(\theta_{n+1},\lambda_n,\eta_n)^\top, \tag{1.49c}$$

where $\nabla_\lambda\mathcal{L}(\theta,\lambda,\eta)^\top=g(\theta)$ and $\nabla_\lambda\mathcal{L}(\theta,\lambda,\eta)^\top=h(\theta)$. The $\max(0,\cdot)$ is a component-wise max operation on the vector.

(continued...)

Index