Appendix C

Optimization Problems and Optimality Conditions

C.1 Unconstrained Optimization

The mathematical model of an (unconstrained) optimization problem can be generally described by a domain or constraint set $\mathcal{D}$ in $\mathbb{R}^n$ and an objective function $f : \mathcal{D} \to \mathbb{R}$ that maps an element of $\mathcal{D}$ to a real value. The optimization problem seeks an optimal solution $\boldsymbol{x}_\star \in \mathcal{D}$ such that the value of f is minimized:

$$f(\boldsymbol{x}_\star) \leq f(\boldsymbol{x}), \quad \text{for all } \boldsymbol{x} \in \mathcal{D}.$$

In particular, if $\mathcal{D} = \mathbb{R}^n$, it is called an unconstrained optimization problem.

Definition C.1.1 (Local and Global Minima)**.** *A variable $\boldsymbol{x}_\star$ is a local minimum of f if there exists a neighborhood $\mathcal{B}_\epsilon(\boldsymbol{x}_\star) \doteq \{\boldsymbol{x} \in \mathcal{D} \mid \|\boldsymbol{x}-\boldsymbol{x}_\star\|_2 < \epsilon\}$ for some $\epsilon > 0$ such that*

$$f(\boldsymbol{x}_\star) \leq f(\boldsymbol{x}), \quad \textit{for all } \boldsymbol{x} \in \mathcal{B}_\epsilon(\boldsymbol{x}_\star).$$

The variable $\boldsymbol{x}_\star$ is a global minimum of f if $\mathcal{B}_\epsilon(\boldsymbol{x}_\star) = \mathcal{D}$. The above local and global minima are said to be strict if the corresponding inequalities are also strict for $\boldsymbol{x} \neq \boldsymbol{x}_\star$.

If the objective function f is differentiable, then conditions for the optimality can be expressed in terms of its derivatives. In particular, if $\boldsymbol{x}_\star$ is a local minimum, then within a small neighborhood $\mathcal{B}_\epsilon(\boldsymbol{x}_\star)$, for any given vector $\boldsymbol{v} \in \mathbb{R}^n$, we have

$$f(\boldsymbol{x}_\star + t \cdot \boldsymbol{v}) \geq f(\boldsymbol{x}_\star)$$

for sufficiently small $t > 0$ such that $t \cdot \boldsymbol{v} \in \mathcal{B}_\epsilon(\mathbf{0})$. Hence we have

$$\lim_{t \to 0} \frac{f(\boldsymbol{x}_\star + t \cdot \boldsymbol{v}) - f(\boldsymbol{x}_\star)}{t} = \nabla f(\boldsymbol{x}_\star)^T \boldsymbol{v} \geq 0.$$

Notice that this must be true for both $\boldsymbol{v}$ and $-\boldsymbol{v}$. Then for the inequality to hold for all $\boldsymbol{v} \in \mathbb{R}^n$, we must have

$$\nabla f(\boldsymbol{x}_\star) = \mathbf{0}. \tag{C.1.1}$$

Definition C.1.2 (Stationary Point)**.** *A point $\boldsymbol{x}_\star$ that satisfies the condition $\nabla f(\boldsymbol{x}_\star) = \mathbf{0}$ is referred to as a stationary point of $f(\boldsymbol{x})$.*

If f is twice continuously differentiable and $\boldsymbol{x}_\star$ is a stationary point with $\nabla f(\boldsymbol{x}_\star) = \mathbf{0}$, we have:

$$f(\boldsymbol{x}_\star + t \cdot \boldsymbol{v}) \approx f(\boldsymbol{x}_\star) + \frac{1}{2} \boldsymbol{v}^T \nabla^2 f(\boldsymbol{x}_\star) \boldsymbol{v} t^2 + o(t^2).$$

If $\boldsymbol{x}_\star$ is a local minimum, we have

$$f(\boldsymbol{x}_\star + t \cdot \boldsymbol{v}) - f(\boldsymbol{x}_\star) \geq 0 \quad \Rightarrow \quad \frac{1}{2} \boldsymbol{v}^T \nabla^2 f(\boldsymbol{x}_\star) \boldsymbol{v} t^2 \geq 0$$

for all $\boldsymbol{v} \in \mathbb{R}^n$. This implies the matrix $\nabla^2 f(\boldsymbol{x}_\star)$ is necessarily positive semi-definite, namely,

$$\nabla^2 f(\boldsymbol{x}_\star) \succeq 0. \tag{C.1.2}$$

It is then not difficult to show the following sufficient condition for local minima:

Proposition C.1.3 (Second-Order Sufficient Optimality Condition)**.** *Let $f : \mathcal{D} \to \mathbb{R}$ be twice continuously differentiable. If $\boldsymbol{x}_\star$ satisfies the conditions*

$$\nabla f(\boldsymbol{x}_\star) = \mathbf{0} \quad \textit{and} \quad \nabla^2 f(\boldsymbol{x}_\star) \succ 0,$$

Then $\boldsymbol{x}_\star$ is a strict local minimum of $f(\boldsymbol{x})$.

In general, a local minimum is not necessarily a global minimum in the domain of $f(\boldsymbol{x})$. Therefore, the global minimum can be found by exhaustively comparing the values of f at all local minima. However, when the objective function f is convex, the following proposition shows that any local minimum is also a global minimum.

Proposition C.1.4 (Global Optimality of Convex Functions)**.** *Let $f : \mathcal{D} \to \mathbb{R}$ be a convex function over convex set $\mathcal{D}$. Then*

1. *A local minimum of f is also a global minimum. Furthermore, if f is strictly convex, then the global minimum, if exists, is unique.*
2. *A point $\boldsymbol{x}_\star \in \mathcal{D}$ is a global minimum of f if $\mathbf{0} \in \partial f(\boldsymbol{x}_\star)$. In the case that f is differentiable, $\nabla f(\boldsymbol{x}_\star) = \mathbf{0}$ implies that $\boldsymbol{x}_\star$ is a global minimum.*

Finally, we note that given an objective function f, a local minimum need not exist. For example, the simple scalar function $f(x) = x$ does not have a minimal value in the domain of real numbers as $\inf_{x\in\mathbb{R}} f(x) = -\infty$. Therefore, a sufficient condition for f to have at least one local minimum is that the set $\{f(\boldsymbol{x})|\boldsymbol{x} \in \mathcal{D}\}$ is bounded below. Alternatively, according to the Weierstrass theorem, if f is continuous and the domain set $\mathcal{D} \subseteq \mathbb{R}^n$ is compact (i.e.closed and bounded), then f has at least one local minimum.

C.2 Constrained Optimization

In the previous section, the constraint set of the optimization problems is assumed to be rather general. However, in most optimization problems considered in this book, the constraints are formulated as equality or inequality conditions. For example, the domain $\mathcal{D} \in \mathbb{R}^n$ of a polyhedron can be specified by a set of equality and inequality conditions. *Lagrange multipliers* are a set of supportive variables to facilitate the derivation of optimality conditions for such constrained optimization problems. Arguably, Lagrange multiplier theory is the most influential theory in constrained optimization. In duality theory that we will discuss in the next section, the same Lagrange multiplier variables are also called *dual variables*, which will play a central role as the optimization variables of the *dual problems*.

First, we consider the optimization problem with equality constraints:

$$\min f(\boldsymbol{x}) \quad \text{subj. to} \quad h_i(\boldsymbol{x}) = 0,\ i = 1, \ldots, m, \tag{C.2.1}$$

where f and each h_i are assumed to be continuously differentiable.[1] Conveniently, we further assume the gradients of the equality conditions at a feasible solution $\boldsymbol{x}$

$$\nabla h_1(\boldsymbol{x}), \nabla h_2(\boldsymbol{x}), \ldots, \nabla h_m(\boldsymbol{x})$$

are linearly independent. Such a solution $\boldsymbol{x}$ is also called *regular*.

The optimality conditions for (C.2.1) can be conveniently derived in terms of the Lagrangian function $L : \mathbb{R}^{n+m} \to \mathbb{R}$ as

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) \doteq f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i h_i(\boldsymbol{x}) = f(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{h}(\boldsymbol{x}) \rangle, \tag{C.2.2}$$

where λ_i are the Lagrange multipliers for the equality conditions, and $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_m]^T \in \mathbb{R}^m$ is the corresponding Lagrange multiplier vector; and for brevity, we denote $\boldsymbol{h} = [h_1, h_2, \ldots, h_m]^T$ as a map from $\mathbb{R}^n$ to $\mathbb{R}^m$.

The basic Lagrange multiplier theory states the following necessary condition for the optimality of a regular solution.

[1] In the main text, we need to generalize to cases when f is not differentiable.

Proposition C.2.1 (Necessary Conditions)**.** *Let $\boldsymbol{x}_\star$ be a local minimum of function $f(\boldsymbol{x})$ subject to $h_i(\boldsymbol{x}) = 0$, $i = 1, \ldots, m$. Further assume $\boldsymbol{x}_\star$ is regular. Then there exists a Lagrange multiplier vector $\boldsymbol{\lambda}_\star = (\lambda_{\star,1}, \lambda_{\star,2}, \ldots, \lambda_{\star,m}) \in \mathbb{R}^m$, such that*

$$\begin{array}{rcl} \nabla_{\boldsymbol{x}} L(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star) & = & \nabla f(\boldsymbol{x}_\star) + \sum_{i=1}^m \lambda_{\star,i} \nabla h_i(\boldsymbol{x}_\star) = 0, \\ \nabla_{\boldsymbol{\lambda}} L(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star) & = & \boldsymbol{h}(\boldsymbol{x}_\star) = 0. \end{array} \tag{C.2.3}$$

Furthermore, if f and $\boldsymbol{h}$ are twice continuously differentiable, we have

$$\begin{array}{rcl} \boldsymbol{v}^T \nabla^2_{\boldsymbol{xx}} L(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star) \boldsymbol{v}^T & = & \boldsymbol{v}^T (\nabla^2 f(\boldsymbol{x}_\star) + \sum_{i=1}^m \lambda_{\star,i} \nabla^2 h_i(\boldsymbol{x}_\star)) \boldsymbol{v}^T \\ & \geq & 0, \quad \forall \boldsymbol{v} : \boldsymbol{v}^T \nabla h_i(\boldsymbol{x}_\star) = 0, \ i = 1, \ldots, m. \end{array} \tag{C.2.4}$$

In (C.2.4), the conditions for vector $\boldsymbol{v} \in \mathbb{R}^n$ that satisfies $\boldsymbol{v}^T \nabla h_i(\boldsymbol{x}_\star) = 0$ can be understood as follows. If we consider a new point $\boldsymbol{x}' = \boldsymbol{x}_\star + \boldsymbol{v}$, due to the fact that $\boldsymbol{v}^T \nabla h_i(\boldsymbol{x}_\star) = 0$, the small variation $\boldsymbol{v}$ will not change the value of $\boldsymbol{h}(\boldsymbol{x}') = 0$. Therefore, we can define

$$V(\boldsymbol{x}_\star) = \{\boldsymbol{v} \mid \boldsymbol{v}^T \nabla h_i(\boldsymbol{x}_\star) = 0, \ i = 1, \ldots, m\}. \tag{C.2.5}$$

as the *subspace of first-order feasible variations*.

In summary, the first-order condition (C.2.3) implies the gradient $\nabla f(\boldsymbol{x}_\star)$ is orthogonal to $V(\boldsymbol{x}_\star)$, which resembles the first-order condition $\nabla f(\boldsymbol{x}_\star) = 0$ in unconstrained optimization. The second-order condition (C.2.4) implies the Hessian of the Lagrangian function $L(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star)$ is positive semidefinite when constrained in $V(\boldsymbol{x}_\star)$.

Proposition C.2.2 (Sufficient Conditions)**.** *Assume f and h are twice continuously differentiable. Let $(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star) \in \mathbb{R}^{n+m}$ satisfy*

$$\begin{array}{rcl} \nabla_{\boldsymbol{x}} L(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star) & = & 0, \\ \nabla_{\boldsymbol{\lambda}} L(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star) & = & 0, \\ \boldsymbol{v}^T \nabla^2_{\boldsymbol{xx}} L(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star) \boldsymbol{v}^T & > & 0 \quad \textit{for all } \boldsymbol{v} \in V(\boldsymbol{x}_\star), \boldsymbol{v} \neq \boldsymbol{0}. \end{array} \tag{C.2.6}$$

Then $\boldsymbol{x}_\star$ is a strict local minimum of $f(\boldsymbol{x})$ subject to $\boldsymbol{h}(\boldsymbol{x}) = 0$.

C.3 Basic Duality Theory

Recall the Lagrangian function for the above equality-constrained optimization problem:

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) \doteq f(\boldsymbol{x}) + \sum_{i=1}^m \lambda_i h_i(\boldsymbol{x}) = f(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{h}(\boldsymbol{x}) \rangle, \tag{C.3.1}$$

where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_m]^T \in \mathbb{R}^m$ are the Lagrangian multipliers.

In duality theory, the vector $\boldsymbol{\lambda}$ is also called the *dual variables* for the so-called *dual function*:

$$q(\boldsymbol{\lambda}) \doteq \inf_{\boldsymbol{x} \in \mathcal{D}} L(\boldsymbol{x}, \boldsymbol{\lambda}). \tag{C.3.2}$$

Correspondingly, $f(\boldsymbol{x})$ is referred to as the *primal function* and $\boldsymbol{x}$ the *primal variables*.

A simple property of the dual function q is that it is a concave function regardless whether the primal problem is convex or not, since q is the point-wise infimum of a family of affine functions with respect to $(\boldsymbol{\lambda})$.

Another important property of the dual function is that $q(\boldsymbol{\lambda})$ is a lower bound of $f(\boldsymbol{x}')$ for any feasible solution $\boldsymbol{x}'$. In particular, $q(\boldsymbol{\lambda})$ is a lower bound of the optimal value $f(\boldsymbol{x}_\star)$. This can be easily verified since for a feasible $\boldsymbol{x}'$ satisfying $\boldsymbol{h}(\boldsymbol{x}') = 0$, we have

$$q(\boldsymbol{\lambda}) = \inf_{\boldsymbol{x}\in\mathcal{D}} f(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{h}(\boldsymbol{x})\rangle \leq \inf_{\boldsymbol{x}\in\mathcal{D}, \boldsymbol{h}(\boldsymbol{x})=0} f(\boldsymbol{x}) \leq f(\boldsymbol{x}').$$

For the dual function $q(\boldsymbol{\lambda})$ to provide a meaningful lower bound for $f(\boldsymbol{x}_\star)$, it is natural to avoid trivial cases when $q(\boldsymbol{\lambda}) = -\infty$. So we normally restrict the domain of the dual function q to:

$$\mathcal{C} \doteq \{\boldsymbol{\lambda} \mid q(\boldsymbol{\lambda}) > -\infty\}. \tag{C.3.3}$$

More specifically, the dual variables $(\boldsymbol{\lambda})$ that satisfy above conditions are called *dual feasible solutions*.

A very useful concept in duality theory is the so-called *duality gap* between the primal and dual functions

$$f(\boldsymbol{x}) - q(\boldsymbol{\lambda}). \tag{C.3.4}$$

Since q is a lower bound of f, the duality gap is always nonnegative (over the set of feasible solutions). More importantly, when the duality gap is zero, namely, there exists a feasible solution $\boldsymbol{x}_\star$ and $\boldsymbol{\lambda}_\star$ such that $f(\boldsymbol{x}_\star) = q(\boldsymbol{\lambda}_\star)$, then $\boldsymbol{x}_\star$ is the optimal primal solution and $\boldsymbol{\lambda}_\star$ is the optimal dual solution.

Since the dual function $q(\boldsymbol{\lambda})$ is a lower bound of the primal function $f(\boldsymbol{x})$, in particular of its minimal value $f(\boldsymbol{x}_\star)$. Naturally, when we want to achieve the best lower-bound estimation of the minimal value, we can consider the following optimization problem in the dual space:

$$\max_{\boldsymbol{\lambda}} q(\boldsymbol{\lambda}). \tag{C.3.5}$$

The problem (C.3.5) is called the *Lagrange dual problem* associated with the original *primal problem* (C.2.1).

Since the optimal solution $q(\boldsymbol{\lambda}_\star)$ is the best lower-bound approximation of the global minimum $f(\boldsymbol{x}_\star)$, the following inequality condition holds trivially:

$$q(\boldsymbol{\lambda}_\star) \leq f(\boldsymbol{x}_\star). \tag{C.3.6}$$

The condition is known as the *weak duality condition*. Furthermore, when the equality can be obtained in (C.3.6), the duality gap between f and q becomes zero, and we say the primal and dual function pair satisfy the *strong duality condition*.

The strong duality condition can be achieved for convex objective functions subject to linear constraints.

Theorem C.3.1 (Strong Duality Theorem)**.** *Let the objective function $f(\boldsymbol{x})$ in (C.2.1) be convex and $\boldsymbol{h}(\boldsymbol{x})$ be linear. If the optimal value $f_\star$ is finite, then the optimal solution for its dual problem exists and there is no duality gap.*

Under the strong duality condition, the minimal value of f can be found by optimizing the dual problem $q(\boldsymbol{\lambda})$, and the optimal primal solution can also be obtained by minimizing the Lagrangian function $L(\boldsymbol{x}, \boldsymbol{\lambda}_\star)$ over $\boldsymbol{x}$. In other words, the optimal $(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star)$ is the saddle point of the Lagrangian function:

$$\max_{\boldsymbol{\lambda}} \min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}). \tag{C.3.7}$$

In the above, we have assumed all functions are differentiable. In this book, we often need to optimize a convex function that is not differentiable and the type of constraints are of the form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$.

Lemma C.3.2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ convex, $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, and let $\boldsymbol{x}_\star$ be some point satisfying $\boldsymbol{A}\boldsymbol{x}_\star = \boldsymbol{y}$. If there exists $\boldsymbol{\nu}$ such that*

$$\boldsymbol{A}^*\boldsymbol{\nu} \in \partial f(\boldsymbol{x}_\star), \tag{C.3.8}$$

then $\boldsymbol{x}_\star$ is a solution to the optimization problem

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \end{array} \tag{C.3.9}$$

Proof. Consider any $\boldsymbol{x}'$ satisfying $\boldsymbol{A}\boldsymbol{x}' = \boldsymbol{y}$. By the subgradient inequality B.3.3,

$$\begin{aligned} f(\boldsymbol{x}') &\geq f(\boldsymbol{x}_\star) + \langle \boldsymbol{A}^*\boldsymbol{\nu}, \boldsymbol{x}' - \boldsymbol{x}_\star \rangle \\ &= f(\boldsymbol{x}_\star) + \langle \boldsymbol{\nu}, \boldsymbol{A}(\boldsymbol{x}' - \boldsymbol{x}_\star) \rangle \\ &= f(\boldsymbol{x}_\star), \end{aligned} \tag{C.3.10}$$

since $\boldsymbol{A}\boldsymbol{x}' = \boldsymbol{A}\boldsymbol{x}_\star$. Thus, $\boldsymbol{x}_\star$ is optimal. □

Appendix D
Methods for Optimization

In this chapter, we review classical approaches to solving optimization problems of the form

$$\min_{\boldsymbol{x}\in\mathcal{D}} f(\boldsymbol{x}), \tag{D.0.1}$$

in which we seek to minimize an objective function f over some domain $\mathcal{D}$. All of the algorithms we describe are *iterative methods* of optimization, which produce a sequence of points

$$\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k, \ldots \tag{D.0.2}$$

starting from some initialization $\boldsymbol{x}_0$. The goal is to generate a sequence $\{\boldsymbol{x}_k\}$ which quickly converges to a minimizer $\boldsymbol{x}_\star$ of f over $\mathcal{D}$. The total time an iterative method requires to produce an acceptable answer depends chiefly on two quantities:

(i) **Per iteration cost**: how long it takes to generate the next point $\boldsymbol{x}_{k+1}$ given the previous points $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_k$.

(ii) **Convergence rate**: how quickly the iterates $\boldsymbol{x}_k$ improve in quality. This dictates how many iterations are required to produce a sufficiently good solution. This may be measured either in terms of the distance of the iterate $\boldsymbol{x}_k$ to a minimizer,

$$\|\boldsymbol{x}_k - \boldsymbol{x}_\star\|, \tag{D.0.3}$$

or in terms of the sub-optimality in objective value,

$$|f(\boldsymbol{x}_k) - f(\boldsymbol{x}_\star)|. \tag{D.0.4}$$

Criteria (i) and (ii) are usually in tension: we can fast convergence rate at the price of very expensive iterations, or we can have very cheap iterations at the price of a relatively slow convergence.

In the era of big data or large models, many practical problems involve optimizing over very large number of model parameters or training over large-scale datasets. Due to computation limitations, we typically can only afford to do fairly simple calculations in each iteration. Hence we are mainly interested in methods that achieve the fastest possible convergence rate out of methods that only work with *first order* information (values of $f(\boldsymbol{x})$ and $\nabla f(\boldsymbol{x})$). Sometimes due to memory limitation and time requirement, we need to store the data and conduct the calculation over many *parallel* processes or a *distributed* network of machines. To reduce communication cost and delay, we often prefer algorithms that are amenable to parallel or distributed implementation and require minimal exchange of data and information across different processes or machines. In this appendix, we sketch basic ideas of some of the most popular and effective techniques that enhance the performance of first order methods, especially those are suitable for solving large-scale problems. We also provide references where the reader can find more complete exposition and analysis of these techniques.

D.1 Gradient Descent

Perhaps the simplest iterative method of optimization is *gradient descent*, also known as the gradient method, which applies to *differentiable* functions $f : \mathbb{R}^n \to \mathbb{R}$. The method comes from the simplest idea that from the current state $\boldsymbol{x}_k$, one would like to take a small step $t \geq 0$ in the direction $\boldsymbol{v} \in \mathbb{R}^n$ to $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t \cdot \boldsymbol{v}$ such that the value of f decreases:

$$f(\boldsymbol{x}_{k+1}) < f(\boldsymbol{x}_k).$$

Since f is differentiable, we know that up to first-order approximation:

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) = f(\boldsymbol{x}_k + t \cdot \boldsymbol{v}) - f(\boldsymbol{x}_k) \approx t \cdot \nabla f(\boldsymbol{x}_k)^T \boldsymbol{v}.$$

The gradient $\nabla f(\boldsymbol{x}_k)$ points in the direction of steepest increase of the objective f; the negative gradient is the direction of steepest decent. So in order for $f(\boldsymbol{x}_{k+1})$ to be smaller than $f(\boldsymbol{x}_k)$, it is natural to take the direction in which the value of f drops the fastest: $\boldsymbol{v} \propto -\nabla f(\boldsymbol{x}_k)$.

Therefore, gradient descent generates its next iterate by stepping in the direction of the negative gradient

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - t_k \nabla f(\boldsymbol{x}_k). \tag{D.1.1}$$

Here, $t_k \geq 0$ is a scalar, often called the *step size*.[1] The step size t_k can either be determined analytically from the properties of the function f, or numerically by performing a *line search*, which produces an approximate solution[2] to the one dimensional problem:

$$\min_{t \geq 0} \; f\left(\boldsymbol{x}_k + t \nabla f(\boldsymbol{x}_k)\right). \tag{D.1.2}$$

Convergence of gradient descent.

A principal virtue of gradient descent is that for many problems, ∇f can be computed efficiently. To understand the overall properties of the method, we need to know how many iterations it requires to obtain a solution of a given desired quality. This depends in turn on the properties of the objective function f.

We begin by assuming that f is a convex, differentiable function, and that the gradient $\nabla f(\boldsymbol{x})$ is L-Lipschitz:

$$\left\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{x}')\right\|_2 \leq L \left\|\boldsymbol{x} - \boldsymbol{x}'\right\|_2, \quad \forall\, \boldsymbol{x}, \boldsymbol{x}'. \tag{D.1.3}$$

This condition states that the gradient does not change too rapidly as we move from point to point. Intuitively, this means that a first-order model for the objective function generated by taking a Taylor expansion at point $\boldsymbol{x}$ will be valid over a relatively large portion of the space. Indeed, it turns out that under these hypotheses, we can take t_k to a uniform $\frac{1}{L}$; smaller L allows larger steps. Moreover, it can be shown that with this choice,

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) &\leq f(\boldsymbol{x}_k) - \tfrac{1}{2L} \left\|\nabla f(\boldsymbol{x}_k)\right\|^2 \\ &\leq f(\boldsymbol{x}_k). \end{aligned} \tag{D.1.4}$$

Thus, with this choice, the gradient method is a *descent method*: it strictly decreases the objective at each iteration, until $\boldsymbol{x}_k$ reaches a minimizer. The following theorem gives an overall control on the rate of convergence, measured in function values:

Theorem D.1.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function with $\nabla f(\boldsymbol{x})$ L-Lipschitz. Let $\mathsf{X}_\star \neq \varnothing$ denote the set of minimizers of f, and $f_\star$ the minimum value of f over $\mathbb{R}^n$. Consider the gradient method with constant step size $t_k = \frac{1}{L}$. Then*

$$f(\boldsymbol{x}_k) - f_\star \leq \frac{L}{2} \frac{\left\|\boldsymbol{x}_0 - \boldsymbol{x}_\star\right\|_2^2}{k}. \tag{D.1.5}$$

Moreover, as $k \to \infty$, $\boldsymbol{x}_k \to \mathsf{X}_\star$.

A proof of this theorem (actually a more generalized version) can be found in Chapter 8, Section 8.2.

---

[1] or the *learning rate* in learning algorithms.

[2] Typically, this is done by *backtracking*: starting from some nominal value of t, we reduce t until ... backtracking, Armijo and Wolfe conditions

Several aspects of this result are worth noting. First, the suboptimality in function values decreases as $1/k$. In particular, as $k \to \infty$, $f(\boldsymbol{x}_k) \to f_\star$. Second, the rate of convergence depends on the Lipschitz constant L – the smaller L is, faster f approaches $f_\star$. Finally, the rate of convergence depends on the distance of the initialization to $\boldsymbol{x}_\star$. A strength of this result is that it is nonasymptotic (the bound works for all k, not just k large) and does not depend on dimension n. For applications, we care not just about function values, but about the quality of the iterates $\boldsymbol{x}_k$. Here, we are guaranteed that $\boldsymbol{x}_k$ approaches $\mathsf{X}_\star$. However, no general, dimension-independent bound on the rate of convergence is known.

D.2 Rates of Convergence and Acceleration

How good the gradient method is? More generally, if we restrict ourselves to relatively simple methods that only use gradient and function value information, what rate can we obtain? This fundamental question motivates the study of lower bounds for the computational efficiency of methods. This requires a model of computation. One simply model for first order methods assumes that at each iteration, the next point $\boldsymbol{x}_{k+1}$ is generated based only on the previous points $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_k$, their function values $f(\boldsymbol{x}_0), \ldots, f(\boldsymbol{x}_k)$, and gradients $\nabla f(\boldsymbol{x}_0), \ldots, \nabla f(\boldsymbol{x}_k)$:

$$f(\boldsymbol{x}_{k+1}) = \mathcal{F}_{k+1}\big(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_k, f(\boldsymbol{x}_0), \ldots, f(\boldsymbol{x}_k), \nabla f(\boldsymbol{x}_0), \ldots, \nabla f(\boldsymbol{x}_k)\big). \tag{D.2.1}$$

This is sometimes referred to as a *black box model*, since the method only accesses the function f through its value and gradient.

It has been shown that:

Theorem D.2.1. *For every L and R, there exists a convex differentiable function f with ∇f L-Lipschitz, and an initial point $\boldsymbol{x}_0$ satisfying $\|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2 \leq R$ such that*

$$f(\boldsymbol{x}_k) - f_\star \geq c\frac{LR^2}{k^2}, \tag{D.2.2}$$

where $c > 0$ is a numerical constant.

This result can be read as saying that for the class of functions with Lipschitz gradients, the best generic rate of convergence that any gradient-like method can achieve is $O(1/k^2)$. Notice that Theorem D.1.1 implies that the gradient method converges at a rate of $O(1/k)$. For large k, this is *much* worse!

Could the gradient method be suboptimal? Figure D.1 shows the behavior of gradient descent on two different problems. The figure plots the level sets $S_\beta = \{\boldsymbol{x} \mid f(\boldsymbol{x}) = \beta\}$ of the objective f as well as the iterates $\boldsymbol{x}_k$. Because the gradient $\nabla f(\boldsymbol{x})$ is orthogonal to the level set containing $\boldsymbol{x}$,

[[**FIGURE HERE**]]

Figure D.1. **Gradient descent on ill-conditioned problems John: show the behavior of the gradient method on two problems – a nearly spherical quadratic and a more ellipsoidal quadratic**

[[**FIGURE HERE**]]

Figure D.2. **The gradient method and heavy ball method.**

the gradient method moves orthogonal to the level sets. At left, we show a function $f(\boldsymbol{x})$ whose level sets are nearly circular. The gradient method makes rapid progress. At right is a function $f(\boldsymbol{x})$ whose level sets are more elongated. The iterates "chatter" repeatedly changing direction and making slow progress towards $\boldsymbol{x}_\star$.

The heavy ball method.

The bad behavior in Figure D.1 can be mitigated by preventing the steps $\boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ from changing direction too rapidly. An intuitive way to accomplish this is to treat the iterate $\boldsymbol{x}_k$ as a particle with some amount of momentum, which causes it to continue moving in the same direction. This suggests an update of the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - t_k \nabla f(\boldsymbol{x}_k) + \beta_k \left(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\right). \tag{D.2.3}$$

Because it treats $\boldsymbol{x}_k$ as a particle with nonzero mass, this method is aptly called the *heavy ball method*. Figure D.2 compares the heavy ball method to the gradient method on an ill-conditioned quadratic. Notice that the heavy ball method takes far fewer iterations to reach the vicinity of $\boldsymbol{x}_\star$.

Nesterov's accelerated method.

Although the heavy ball method improves over the gradient method, its worst case rate of convergence is still $O(1/k)$. However, by using momentum in a clever way, it *is* possible to achieve a better rate of convergence of $O(1/k^2)$, which matches the lower bound in Theorem D.2.1. This means, perhaps surprisingly, that there is a gradient-like method that is fundamentally better than gradient descent!

The method that achieves this optimal rate is known as *Nesterov's accelerated gradient method*. Strictly speaking, it is not a momentum method. Rather, it uses two sequences of iterates $\boldsymbol{x}_k$ and $\boldsymbol{p}_k$. The auxiliary point $\boldsymbol{p}_k$ is extrapolated from $\boldsymbol{x}_k$ in a form similar to that in the heavy ball method:

$$\boldsymbol{p}_{k+1} \doteq \boldsymbol{x}_k + \beta_k\big(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\big).$$

At each iteration, we move to this new point, compute the gradient at this point, and descend from it (instead of $\boldsymbol{x}_k$):

$$\boldsymbol{x}_{k+1} = \boldsymbol{p}_{k+1} - \alpha \nabla f(\boldsymbol{p}_{k+1}). \tag{D.2.4}$$

As we will show in Section 8.3 of Chapter 8, with properly chosen weights β_k and α, the gradient method is indeed accelerated and can achieve the optimal convergence rate of $O(1/k^2)$, for the class of functions with Lipschitz gradients,.

Theorem D.2.2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function with $\nabla f(\boldsymbol{x})$ being L-Lipschitz. Let $\mathsf{X}_\star \neq \varnothing$ denote the set of minimizers of f and $f_\star$ the minimum value of f over $\mathbb{R}^n$. The iterates $\boldsymbol{x}_k$ produced by the accelerated gradient method satisfy*

$$f(\boldsymbol{x}_k) - f_\star \leq \frac{L}{2} \frac{\|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2}{(k+1)^2}. \tag{D.2.5}$$

Moreover, as $k \to \infty$, $\boldsymbol{x}_k \to \boldsymbol{x}_\star$.

Strongly convex functions.

Notice that Theorem D.2.1 characterizes the best possible rate of convergence for gradient-like methods for the class of functions with Lipschitz gradients; and Theorem D.2.2 states that this rate can be achieved with the accelerated gradient methods. Nevertheless, this does not means this is the best one can do for more restricted classes of functions. If, in addition to Lipschitz gradients, the functions satisfy additional nice properties (such as strongly convex defined in Appendix B), one can show that gradient-like methods converge at a *linear rate* [Boyd and Vandenberghe, 2004].

Theorem D.2.3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable strongly convex function with constant μ and $\nabla f(\boldsymbol{x})$ being L-Lipschitz. Let $f_\star$ be the minimum value of f over $\mathbb{R}^n$. Then the iterates $\boldsymbol{x}_k$ produced by the gradient-descent $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - t\nabla f(\boldsymbol{x}_k)$ with $t = \frac{2}{L+\mu}$ satisfy*

$$f(\boldsymbol{x}_k) - f_\star \leq \frac{L}{2} e^{-\alpha k} \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2 \tag{D.2.6}$$

for some constant $\alpha > 0$.

That is, $f(\boldsymbol{x}_k) - f_\star$ converges to zero exponentially in the order of $O(e^{-\alpha k})$, much faster than $O(1/k^2)$. In this book, the class of functions that we often encounter are not necessarily strongly convex. Nevertheless, they may satisfy certain weaker notion of strong convexity, known as *restricted strong convexity*. We will see that under such conditions, one may also expect gradient-like methods to achieve linear rate of convergence.

Nondifferentiable functions.

The main assumption of gradient descent methods is that the objective function $f(\boldsymbol{x})$ is differentiable in $\boldsymbol{x}$. In this book, we often need to minimize

functions that are not everywhere differentiable, such as functions involving the ℓ^1 norm $\|\boldsymbol{x}\|_1$. In such cases, we need to generalize the notion of gradient to "subgradients" (see Definition 2.3.4 in Chapter 2). Essentially, subgradients at a point $\boldsymbol{x}$ is the set of vectors $\boldsymbol{u} \in \mathbb{R}^n$ such that

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \boldsymbol{u}, \boldsymbol{y} - \boldsymbol{x} \rangle, \quad \forall \boldsymbol{y}.$$

We often denote the set of subgradients as $\partial f(\boldsymbol{x})$. To minimize such a function $f(\boldsymbol{x})$, we may generalize the gradient descent method by replacing the gradient $\nabla f(\boldsymbol{x})$ with any subgradient:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - t_k \boldsymbol{g}_k, \quad \boldsymbol{g}_k \in \partial f(\boldsymbol{x}).$$

A main disadvantage of such subgradient descent methods is their relatively poor convergence rate. In general, the convergence rate of subgradient descent for non-smooth objective functions is

$$f(\boldsymbol{x}_k) - f_\star = O\left(1/\sqrt{k}\right).$$

The reader can refer to [Nemirovski, 1995, Nemirovski, 2007, Nesterov, 2003b] for more detailed analysis of subgradient descent algorithms.

Nevertheless, as we will see in Chapter 8, in many of our problems, the objective function $f(\boldsymbol{x})$ is of the form $f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})$ with f_1 being smooth and f_2 nonsmooth. If for the nonsmooth part f_2, the so called *proximal operator*:

$$\min_{\boldsymbol{x}} f_2(\boldsymbol{x}) + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{w}\|_2^2 \tag{D.2.7}$$

has a closed-form solution or can be solved efficiently, then the subgradient descent method can be properly modified so that it would enjoy the same convergence rate as the smooth case. See the *proximal gradient* method in Section 8.2 of Chapter 8.

Constrained optimization.

It is very common in practice that we want to minimize a function $f(\boldsymbol{x})$ while the desired solution $\boldsymbol{x}_\star$ is constrained to some subset $C \subset \mathbb{R}^n$:

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in C. \end{array} \tag{D.2.8}$$

Solutions in the subset C are called *feasible* solutions. Notice that if we still apply the gradient descent method to minimize $f(\boldsymbol{x})$. Then, at each descent iteration $\boldsymbol{p}_{k+1} = \boldsymbol{x}_k - t_k \nabla f(\boldsymbol{x}_k)$, even if $\boldsymbol{x}_k$ is feasible, the new state $\boldsymbol{p}_{k+1}$ may step outside of the constrained set: $\boldsymbol{p}_{k+1} \notin C$. A natural and simple fix to this issue is to "project" $\boldsymbol{p}_{k+1}$ back to the set C:

$$\boldsymbol{x}_{k+1} = \mathcal{P}_C[\boldsymbol{p}_{k+1}] = \arg\min_{\boldsymbol{x} \in C} \tfrac{1}{2} \left\|\boldsymbol{x} - \boldsymbol{p}_{k+1}\right\|_2^2 \tag{D.2.9}$$

where $\boldsymbol{x}_{k+1}$ is the point in C closest to $\boldsymbol{p}_{k+1}$. This will ensure the new iterate $\boldsymbol{x}_{k+1}$ is always feasible. This method is called *projected gradient descent*,

and we use it to provide a simplest algorithm for minimizing the ℓ^1 norm in Chapter 2. This simple method is also the inspiration for other first-order constrained optimization methods such as the *Frank-Wolfe* method introduced in Section 8.6 of Chapter 8.

One disadvantage of such projected gradient descent methods is their relatively poor convergence rate. In the case the constraints are equality constraints: $C = \{\boldsymbol{x} \mid \boldsymbol{h}(\boldsymbol{x}) = 0\}$, one could try to convert the constrained optimization

$$\text{minimize} \quad f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{h}(\boldsymbol{x}) = 0 \tag{D.2.10}$$

to an unconstrained one by penalizing any deviation of $\boldsymbol{h}(\boldsymbol{x})$ from 0:

$$\min f(\boldsymbol{x}) + \frac{\mu}{2}\|\boldsymbol{h}(\boldsymbol{x})\|_2^2. \tag{D.2.11}$$

This is known as the *penalty method*. One can show that as $\mu \to +\infty$, the solution to the unconstrained optimization approaches that of the constrained one. However, in practice, as μ becomes large, the unconstrained problem becomes increasingly harder to solve as its gradient Lipschitz becomes increasingly large. See Section 8.4 of Chapter 8 for an example.

As we have discussed in Appendix C, another way to convert the constrained optimization problem is through the Lagrangian formulation. The optimal (feasible) solution $\boldsymbol{x}_\star$ to the above constrained optimization is also the optimal solution $(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star)$ to the unconstrained optimization:

$$\max_{\boldsymbol{\lambda}} \min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}) \tag{D.2.12}$$

where the Lagrangian function is defined as

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) \doteq f(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{h}(\boldsymbol{x}) \rangle.$$

It is natural to consider solving the above min-max problem through the following alternating optimization scheme:

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}_k), \tag{D.2.13}$$

$$\boldsymbol{\lambda}_{k+1} = \arg\max_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}). \tag{D.2.14}$$

Although the saddle point of the Lagrangian is the desired optimal solution, there is no guarantee that each step of the above iteration would produce feasible iterates. As we see in Section 8.4 of Chapter 8, even for some simple problems, the above subproblems might fail to have a solution (the value of the objective function can be unbounded).

To remedy this problem, one could augment the Lagrangian $L(\boldsymbol{x}, \boldsymbol{\lambda})$ with an extra quadratic penalty term for the constraint:

$$L_\mu(\boldsymbol{x}, \boldsymbol{\lambda}) \doteq f(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{h}(\boldsymbol{x}) \rangle + \frac{\mu}{2}\|\boldsymbol{h}(\boldsymbol{x})\|_2^2,$$

which is known as the *Augmented Lagrangian*. As we will see in Section 8.4 of Chapter 8, the augmented Lagrangian leads to much better conditioned

subproblems for the alternating scheme:

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} \mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{\lambda}_k), \tag{D.2.15}$$

$$\boldsymbol{\lambda}_{k+1} = \arg\max_{\boldsymbol{\lambda}} \mathcal{L}_\mu(\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}), \tag{D.2.16}$$

and the sequence of iterates $\{(\boldsymbol{x}_k, \boldsymbol{\lambda}_k)\}$ typically converge to the desired optimal solution $(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star)$ for a properly chosen μ or a sequence $\{\mu_k\}$.

D.3 Block Coordinate Descent and Alternating Direction Method of Multipliers

In many optimization problems we may encounter in practice, the dimension of $\boldsymbol{x}$ could be so high that we might not even afford to conduct gradient descent to minimize $f(\boldsymbol{x})$ for all the variables together. Very often the objective function $f(\boldsymbol{x})$ has certain decomposable structures such as:

$$\min f(\boldsymbol{x}) = \sum_{i=1}^{m} f_i(\boldsymbol{x}^i). \tag{D.3.1}$$

For example, the ℓ^1-norm function $\|\boldsymbol{x}\|_1 = \sum_{i=1}^n |x_i|$ is such a decomposable function. In such cases, we may conduct the so-called *block coordinate descent* to take advantage of such decomposable structures by iteratively minimizing the objective function with respect to one block of variables at a time.

More specifically, assume the domain $\mathcal{D}$ can be written as a Cartesian product

$$\mathcal{D} = \mathcal{D}_1 \times \mathcal{D}_2 \times \cdots \times \mathcal{D}_m,$$

where each $\mathcal{D}_i \subseteq \mathbb{R}^{n_i}$, $n_1 + n_2 + \cdots + n_m = n$. The variables can be also partitioned into m blocks as $\boldsymbol{x} = (\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^m) \in \mathbb{R}^n$ with each $\boldsymbol{x}^i \in \mathcal{D}_i$. The block coordinate descent scheme proceeds as follows:

1. Initialize $\boldsymbol{x}_0 = (\boldsymbol{x}_0^1, \boldsymbol{x}_0^2, \ldots, \boldsymbol{x}_0^m)$.
2. In the k-th iteration, for every $i = 1, \ldots, m$,
$$\boldsymbol{x}_k^i = \arg\min_{\bar{\boldsymbol{x}} \in \mathcal{D}_i} f(\boldsymbol{x}_k^1, \ldots, \boldsymbol{x}_k^{i-1}, \bar{\boldsymbol{x}}, \boldsymbol{x}_{k-1}^{i+1}, \ldots, \boldsymbol{x}_{k-1}^m).$$
3. Repeat Step 2 until the solution converges.

In the literature, the convergence of block coordinate descent methods can be proven under different conditions. A most natural condition is when the objective function $f(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{i-1}, \boldsymbol{x}^i, \boldsymbol{x}^{i+1}, \ldots, \boldsymbol{x}^m)$ is *strictly convex* with respect to each block $\boldsymbol{x}_i$. This guarantees the minimal solution $\boldsymbol{x}_\star^i$ is also unique. For a more detailed discussion about conditions under which such methods converge, the reader is referred to [Bertsekas, 2003].

In compressive sensing or statistical learning,[3] very often we need to deal with an objective function $f(\boldsymbol{x})$ that is composed of multiple terms:

$$f(\boldsymbol{x}) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}) + \cdots + f_m(\boldsymbol{x}). \tag{D.3.2}$$

To obtain more scalable algorithms such that we can optimize each term in a parallel or distributed fashion, we could rewrite this problem in terms of a set of local variables $\boldsymbol{x}^i \in \mathbb{R}^n$ and one global variable $\boldsymbol{z}$:

$$\text{minimize} \quad \sum_{i=1}^{m} f_i(\boldsymbol{x}^i) \quad \text{subject to} \quad \boldsymbol{x}^i = \boldsymbol{z}, \quad i = 1, \ldots, m. \tag{D.3.3}$$

In the literature, this is also known as the *consensus optimization*. To solve such a constrained optimization problem, we could apply the above block descent method to its augment Lagrangian:

$$L(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^m, \boldsymbol{z}, \boldsymbol{\lambda}) = \sum_{i=1}^{m} f_i(\boldsymbol{x}^i) + \langle \boldsymbol{\lambda}^i, \boldsymbol{x}^i - \boldsymbol{z} \rangle + \frac{\mu}{2} \|\boldsymbol{x}^i - \boldsymbol{z}\|_2^2. \tag{D.3.4}$$

This leads to the following iterative process:

$$\begin{aligned}
\boldsymbol{x}_{k+1}^i &= \arg\min_{\boldsymbol{x}^i} f_i(\boldsymbol{x}^i) + \langle \boldsymbol{\lambda}^i, \boldsymbol{x}^i - \boldsymbol{z} \rangle + \frac{\mu}{2} \|\boldsymbol{x}^i - \boldsymbol{z}\|_2^2, \\
\boldsymbol{z}_{k+1} &= \frac{1}{m} \sum_{i=1}^{m} \big(\boldsymbol{x}_{k+1}^i + \frac{1}{\mu} \boldsymbol{\lambda}_k^i \big), \\
\boldsymbol{\lambda}_{k+1}^i &= \boldsymbol{\lambda}_k^i + \mu \big(\boldsymbol{x}_{k+1}^i - \boldsymbol{z}_{k+1} \big).
\end{aligned}$$

This is known as the *Alternating Direction Method of Multipliers* (ADMM). Notice that the above scheme is rather amenable to distributed implementation as each local process can solve in parallel a subproblem for $\boldsymbol{x}^i$ and then share the information through the common variable $\boldsymbol{z}$. In Chapter 8, we will study the ADMM scheme for the case with $m = 2$ in great detail. For a more detailed exposition of ADMM and more general variants, the reader may refer to [Boyd et al., 2011b].

D.4 Nonconvex Problems

nonconvex problems – convergence to a stationary point? avoid saddle points? trust region methods?

[3] Say training a deep neural networks over a very large set of training samples, where $\boldsymbol{x}$ are the network parameters.