IAS/Park City Mathematics Series
Volume 00, Pages 000–000
S 1079-5634(XX)0000-0

Optimization Algorithms for Data Analysis

Stephen J. Wright

Contents

2010 *Mathematics Subject Classification.* Primary 14Dxx; Secondary 14Dxx.
Key words and phrases. Park City Mathematics Institute.

1. Introduction

In this article, we consider algorithms for solving smooth optimization problems, possibly with simple constraints or structured nonsmooth regularizers. One such canonical formulation is

$$\min_{x \in \mathbb{R}^n} f(x), \tag{1.0.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ has at least Lipschitz continuous gradients. Additional assumptions about f, such as convexity and Lipschitz continuity of the Hessian, are introduced as needed. Another formulation we consider is

$$\min_{x \in \mathbb{R}^n} f(x) + \lambda \psi(x), \tag{1.0.2}$$

where f is as in (1.0.1), $\psi : \mathbb{R}^n \to \mathbb{R}$ is a function that is usually convex and usually nonsmooth, and $\lambda \geqslant 0$ is a regularization parameter.[1] We refer to (1.0.2) as a *regularized* minimization problem because the presence of the term involving ψ induces certain structural properties on the solution, that make it more desirable or plausible in the context of the application. We describe iterative algorithms that generate a sequence $\{x^k\}_{k=0,1,2,\ldots}$ of points that, in the case of convex objective functions, converges to the set of solutions. (Some algorithms also generate other "auxiliary" sequences of iterates.)

We are motivated to study problems of the forms (1.0.1) and (1.0.2) by their ubiquity in data analysis applications. Accordingly, Section 2 describes some canonical problems in data analysis and their formulation as optimization problems. After some preliminaries in Section 3, we describe in Section 4 algorithms that take step based on the gradients $\nabla f(x^k)$. Extensions of these methods to

[1] A set S is said to be *convex* if for any pair of points $z', z'' \in S$, we have that $\alpha z' + (1-\alpha) z'' \in S$ for all $\alpha \in [0,1]$. A function $\phi : \mathbb{R}^n \to \mathbb{R}$ is convex if $\phi(\alpha z' + (1-\alpha) z'') \leqslant \alpha \phi(z') + (1-\alpha)\phi(z'')$ for all z', z'' in the (convex) domain of ϕ and all $\alpha \in [0,1]$.

the case (1.0.2) of regularized objectives are described in Section 5. Section 6 describes accelerated gradient methods, which achieve better worst-case complexity than basic gradient methods, while still only using first-derivative information. We discuss Newton's method in Section 7, outlining variants that can guarantee convergence to points that approximately satisfy second-order conditions for a local minimizer of a smooth nonconvex function.

1.1. Omissions Our approach throughout is to give a concise description of some of the most important algorithmic tools for smooth nonlinear optimization and regularized optimization, along with the basic convergence theory for each. (In any given context, we mean by "smooth" that the function is differentiable as many times as is necessary for the discussion to make sense.) In most cases, the theory is elementary enough to include here in its entirety. In the few remaining cases, we provide citations to works in which complete proofs can be found.

Although we allow nonsmoothness in the regularization term in (1.0.2), we do not cover subgradient methods or mirror descent explicitly in this chapter. We also do not discuss stochastic gradient methods, a class of methods that is central to modern machine learning. All these topics are discussed in the contribution of John Duchi to the current volume [22]. Other omissions include the following.

- Coordinate descent methods; see [47] for a recent review.
- Augmented Lagrangian methods, including alternating direction methods of multipliers (ADMM) [23]. The review [5] remains a good reference for the latter topic, especially as it applies to problems from data analysis.
- Semidefinite programming (see [43,45]) and conic optimization (see [6]).
- Methods tailored specifically to linear or quadratic programming, such as the simplex method or interior-point methods (see [46] for a discussion of the latter).
- Quasi-Newton methods, which modify Newton's method by approximating the Hessian or its inverse, thus attaining attractive theoretical and practical performance without using any second-derivative information. For a discussion of these methods, see [36, Chapter 6]. One important method of this class, which is useful in data analysis and many other large-scale problems, is the limited-memory method L-BFGS [30]; see also [36, Section 7.2].

1.2. Notation Our notational conventions in this chapter are as follows. We use upper-case Roman characters (A, L, R, and so on) for matrices and lower-case Roman (x, v, u, and so on) for vectors. (Vectors are assumed to be *column* vectors.) Transposes are indicated by a superscript "T." Elements of matrices and vectors are indicated by subscripts, for example, A_{ij} and x_j. Iteration numbers are indicated by superscripts, for example, x^k. We denote the set of real numbers by $\mathbb{R}$, so that $\mathbb{R}^n$ denotes the Euclidean space of dimension n. The set of symmetric real $n \times n$ matrices is denoted by $\mathbb{SR}^{n\times n}$. Real scalars are usually denoted by

Greek characters, for example, α, β, and so on, though in deference to convention, we sometimes use Roman capitals (for example, L for the Lipschitz constant of a gradient). Where vector norms appear, the type of norm in use is indicated by a subscript (for example $\|x\|_1$), except that when no subscript appears, the Euclidean norm $\|\cdot\|_2$ is assumed. Matrix norms are defined where first used.

2. Optimization Formulations of Data Analysis Problems

In this section, we describe briefly some representative problems in data analysis and machine learning, emphasizing their formulation as optimization problems. Our list is by no means exhaustive. In many cases, there are a number of different ways to formulate a given application as an optimization problem. We do not try to describe all of them. But our list here gives a flavor of the interface between data analysis and optimization.

2.1. Setup Practical data sets are often extremely messy. Data may be mislabeled, noisy, incomplete, or otherwise corrupted. Much of the hard work in data analysis is done by professionals, familiar with the underlying applications, who "clean" the data and prepare it for analysis, while being careful not to change the essential properties that they wish to discern from the analysis. Dasu and Johnson [19] claim out that "80% of data analysis is spent on the process of cleaning and preparing the data." We do not discuss this aspect of the process, focusing instead on the part of the data analysis pipeline in which the problem is formulated and solved.

The data set in a typical analysis problem consists of m objects:

$$D := \{(a_j, y_j),\ j = 1, 2, \ldots, m\}, \tag{2.1.1}$$

where a_j is a vector (or matrix) of *features* and y_j is an *observation* or *label*. (Each pair (a_j, y_j) has the same size and shape for all $j = 1, 2, \ldots, m$.) The analysis task then consists of discovering a function ϕ such that $\phi(a_j) \approx y_j$ holds for most $j = 1, 2, \ldots, m$. The process of discovering the mapping ϕ is often called "learning" or "training."

The function ϕ is often defined in terms of a vector or matrix of parameters, which we denote by x or X. (Other notation also appears below.) With these parametrizations, the problem of identifying ϕ becomes a data-fitting problem: "Find the parameters x defining ϕ such that $\phi(a_j) \approx y_j$, $j = 1, 2, \ldots, m$ in some optimal sense." Once we come up with a definition of the term "optimal," we have an optimization problem. Many such optimization formulations have objective functions of the "summation" type

$$L_D(x) := \sum_{j=1}^{m} \ell(a_j, y_j; x), \tag{2.1.2}$$

where the jth term $\ell(a_j, y_j; x)$ is a measure of the mismatch between $\phi(a_j)$ and y_j, and x is the vector of parameters that determines ϕ.

One use of ϕ is to make predictions about future data items. Given another previously unseen item of data $\hat{a}$ of the same type as a_j, $j = 1, 2, \ldots, m$, we predict that the label $\hat{y}$ associated with $\hat{a}$ would be $\phi(\hat{a})$. The mapping may also expose other structure and properties in the data set. For example, it may reveal that only a small fraction of the features in a_j are needed to reliably predict the label y_j. (This is known as *feature selection*.) The function ϕ or its parameter x may also reveal important structure in the data. For example, X could reveal a low-dimensional subspace that contains most of the a_j, or X could reveal a matrix with particular structure (low-rank, sparse) such that observations of X prompted by the feature vectors a_j yield results close to y_j.

Examples of labels y_j include the following.

- A real number, leading to a *regression* problem.
- A label, say $y_j \in \{1, 2, \ldots, M\}$ indicating that a_j belongs to one of M classes. This is a *classification* problem. We have $M = 2$ for binary classification and $M > 2$ for multiclass classification.
- Null. Some problems only have feature vectors a_j and no labels. In this case, the data analysis task may consist of grouping the a_j into clusters (where the vectors within each cluster are deemed to be functionally similar), or identifying a low-dimensional subspace (or a collection of low-dimensional subspaces) that approximately contains the a_j. Such problems require the labels y_j to be learned, alongside the function ϕ. For example, in a clustering problem, y_j could represent the cluster to which a_j is assigned.

Even after cleaning and preparation, the setup above may contain many complications that need to be dealt with in formulating the problem in rigorous mathematical terms. The quantities (a_j, y_j) may contain noise, or may be otherwise corrupted. We would like the mapping ϕ to be robust to such errors. There may be *missing data*: parts of the vectors a_j may be missing, or we may not know all the labels y_j. The data may be arriving in *streaming* fashion rather than being available all at once. In this case, we would learn ϕ in an *online* fashion.

One particular consideration is that we wish to avoid *overfitting* the model to the data set D in (2.1.1). The particular data set D available to us can often be thought of as a finite sample drawn from some underlying larger (often infinite) collection of data, and we wish the function ϕ to perform well on the unobserved data points as well as the observed subset D. In other words, we want ϕ to be not too sensitive to the particular sample D that is used to define empirical objective functions such as (2.1.2). The optimization formulation can be modified in various ways to achieve this goal, by the inclusion of constraints or penalty terms that limit some measure of "complexity" of the function (such techniques are called *generalization* or *regularization*). Another approach is to terminate the optimization algorithm early, the rationale being that overfitting occurs mainly in the later stages of the optimization process.

2.2. Least Squares Probably the oldest and best-known data analysis problem is linear least squares. Here, the data points (a_j, y_j) lie in $\mathbb{R}^n \times \mathbb{R}$, and we solve

$$\min_x \frac{1}{2m} \sum_{j=1}^{m} (a_j^T x - y_j)^2 = \frac{1}{2m} \|Ax - y\|_2^2, \tag{2.2.1}$$

where A is the matrix whose rows are a_j^T, $j = 1, 2, \ldots, m$ and $y = (y_1, y_2, \ldots, y_m)^T$. In the terminology above, the function ϕ is defined by $\phi(a) := a^T x$. (We could also introduce a nonzero intercept by adding an extra parameter $\beta \in \mathbb{R}$ and defining $\phi(a) := a^T x + \beta$.) This formulation can be motivated statistically, as a maximum-likelihood estimate of x when the observations y_j are exact but for i.i.d. Gaussian noise. Randomized linear algebra methods for large-scale instances of this problem are discussed in Section 5 of the lectures of Drineas and Mahoney [20] in this volume.

Various modifications of (2.2.1) impose desirable structure on x and hence on ϕ. For example, Tikhonov regularization with a squared ℓ_2-norm, which is

$$\min_x \frac{1}{2m} \|Ax - y\|_2^2 + \lambda \|x\|_2^2, \quad \text{for some parameter } \lambda > 0,$$

yields a solution x with less sensitivity to perturbations in the data (a_j, y_j). The LASSO formulation

$$\min_x \frac{1}{2m} \|Ax - y\|_2^2 + \lambda \|x\|_1 \tag{2.2.2}$$

tends to yield solutions x that are sparse, that is, containing relatively few nonzero components [42]. This formulation performs feature selection: The locations of the nonzero components in x reveal those components of a_j that are instrumental in determining the observation y_j. Besides its statistical appeal — predictors that depend on few features are potentially simpler and more comprehensible than those depending on many features — feature selection has practical appeal in making predictions about future data. Rather than gathering all components of a new data vector $\hat{a}$, we need to find only the "selected" features, since only these are needed to make a prediction. The LASSO formulation (2.2.2) is an important prototype for many problems in data analysis, in that it involves a regularization term $\lambda \|x\|_1$ that is nonsmooth and convex, but with relatively simple structure that can potentially be exploited by algorithms.

2.3. Matrix Completion Matrix completion is in one sense a natural extension of least-squares to problems in which the data a_j are naturally represented as matrices rather than vectors. Changing notation slightly, we suppose that each A_j is an $n \times p$ matrix, and we seek another $n \times p$ matrix X that solves

$$\min_X \frac{1}{2m} \sum_{j=1}^{m} (\langle A_j, X \rangle - y_j)^2, \tag{2.3.1}$$

where $\langle A, B \rangle := \text{trace}(A^T B)$. Here we can think of the A_j as "probing" the unknown matrix X. Commonly considered types of observations are random linear

combinations (where the elements of A_j are selected i.i.d. from some distribution) or single-element observations (in which each A_j has 1 in a single location and zeros elsewhere). A regularized version of (2.3.1), leading to solutions X that are low-rank, is

$$\min_X \frac{1}{2m}\sum_{j=1}^{m}(\langle A_j, X\rangle - y_j)^2 + \lambda\|X\|_*, \tag{2.3.2}$$

where $\|X\|_*$ is the nuclear norm, which is the sum of singular values of X [39]. The nuclear norm plays a role analogous to the ℓ_1 norm in (2.2.2). Although the nuclear norm is a somewhat complex nonsmooth function, it is at least convex, so that the formulation (2.3.2) is also convex. This formulation can be shown to yield a statistically valid solution when the true X is low-rank and the observation matrices A_j satisfy a "restricted isometry" property, commonly satisfied by random matrices, but not by matrices with just one nonzero element. The formulation is also valid in a different context, in which the true X is incoherent (roughly speaking, it does not have a few elements that are much larger than the others), and the observations A_j are of single elements [10].

In another form of regularization, the matrix X is represented explicitly as a product of two "thin" matrices L and R, where $L \in \mathbb{R}^{n\times r}$ and $R \in \mathbb{R}^{p\times r}$, with $r \ll \min(n,p)$. We set $X = LR^T$ in (2.3.1) and solve

$$\min_{L,R} \frac{1}{2m}\sum_{j=1}^{m}(\langle A_j, LR^T\rangle - y_j)^2. \tag{2.3.3}$$

In this formulation, the rank r is "hard-wired" into the definition of X, so there is no need to include a regularizing term. This formulation is also typically much more compact than (2.3.2); the total number of elements in (L,R) is $(n+p)r$, which is much less than np. A disadvantage is that it is nonconvex. An active line of current research, pioneered in [9] and also drawing on statistical sources, shows that the nonconvexity is benign in many situations, and that under certain assumptions on the data (A_j, y_j), $j = 1,2,\ldots,m$ and careful choice of algorithmic strategy, good solutions can be obtained from the formulation (2.3.3). A clue to this good behavior is that although this formulation is nonconvex, it is in some sense an approximation to a tractable problem: If we have a complete observation of X, then a rank-r approximation can be found by performing a singular value decomposition of X, and defining L and R in terms of the r leading left and right singular vectors.

2.4. Nonnegative Matrix Factorization Some applications in computer vision, chemometrics, and document clustering require us to find factors L and R like those in (2.3.3) in which all elements are nonnegative. If the full matrix $Y \in \mathbb{R}^{n\times p}$ is observed, this problem has the form

$$\min_{L,R} \|LR^T - Y\|_F^2, \quad \text{subject to } L \geqslant 0,\ R \geqslant 0.$$

2.5. Sparse Inverse Covariance Estimation In this problem, the labels y_j are null, and the vectors $a_j \in \mathbb{R}^n$ are viewed as independent observations of a random vector $A \in \mathbb{R}^n$, which has zero mean. The sample covariance matrix constructed from these observations is

$$S = \frac{1}{m-1} \sum_{j=1}^{m} a_j a_j^T.$$

The element S_{il} is an estimate of the covariance between the ith and lth elements of the random variable vector A. Our interest is in calculating an estimate X of the *inverse* covariance matrix that is *sparse*. The structure of X yields important information about A. In particular, if $X_{il} = 0$, we can conclude that the i and l components of A are *conditionally independent*. (That is, they are independent given knowledge of the values of the other $n-2$ components of A.) Stated another way, the nonzero locations in X indicate the arcs in the dependency graph whose nodes correspond to the n components of A.

One optimization formulation that has been proposed for estimating the inverse sparse covariance matrix X is the following:

$$\min_{X \in \mathbb{SR}^{n\times n},\, X \succeq 0} \langle S, X\rangle - \log\det(X) + \lambda \|X\|_1, \tag{2.5.1}$$

where $\mathbb{SR}^{n\times n}$ is the set of $n \times n$ symmetric matrices, $X \succeq 0$ indicates that X is positive definite, and $\|X\|_1 := \sum_{i,l=1}^{n} |X_{il}|$ (see [17, 25]).

2.6. Sparse Principal Components The setup for this problem is similar to the previous section, in that we have a sample covariance matrix S that is estimated from a number of observations of some underlying random vector. The *principal components* of this matrix are the eigenvectors corresponding to the largest eigenvalues. It is often of interest to find *sparse* principal components, approximations to the leading eigenvectors that also contain few nonzeros. An explicit optimization formulation of this problem is

$$\max_{v \in \mathbb{R}^n} v^T S v \quad \text{s.t. } \|v\|_2 = 1, \;\; \|v\|_0 \leqslant k, \tag{2.6.1}$$

where $\|\cdot\|_0$ indicates the cardinality of v (that is, the number of nonzeros in v) and k is a user-defined parameter indicating a bound on the cardinality of v. The problem (2.6.1) is NP-hard, so exact formulations (for example, as a quadratic program with binary variables) are intractable. We consider instead a relaxation, due to [18], which replaces vv^T by a positive semidefinite proxy $M \in \mathbb{SR}^{n\times n}$:

$$\max_{M \in \mathbb{SR}^{n\times n}} \langle S, M\rangle \quad \text{s.t. } M \succeq 0,\; \langle I, M\rangle = 1,\; \|M\|_1 \leqslant \rho, \tag{2.6.2}$$

for some parameter $\rho > 0$ that can be adjusted to attain the desired sparsity. This formulation is a convex optimization problem, in fact, a semidefinite programming problem.

This formulation can be generalized to find the leading $r > 1$ sparse principal components. Ideally, we would obtain these from a matrix $V \in \mathbb{R}^{n\times r}$ whose

columns are mutually orthogonal and have at most k nonzeros each. We can write a convex relaxation of this problem, once again a semidefinite program, as

$$\max_{M \in \mathbb{SR}^{n\times n}} \langle S, M\rangle \quad \text{s.t. } 0 \preceq M \preceq I,\ \langle I, M\rangle = 1,\ \|M\|_1 \leqslant \rho\,. \tag{2.6.3}$$

A more compact (but nonconvex) formulation is

$$\max_{F\in\mathbb{R}^{n\times r}} \langle S, FF^T\rangle \quad \text{s.t. } \|F\|_2 \leqslant 1,\ \|F\|_{2,1} \leqslant \bar{R},$$

where $\|F\|_{2,1} := \sum_{i=1}^n \|F_{i\cdot}\|_2$ [15]. The latter regularization term is often called a "group-sparse" or "group-LASSO" regularizer. (An early use of this type of regularizer was described in [44].)

2.7. Sparse Plus Low-Rank Matrix Decomposition Another useful paradigm is to decompose a partly or fully observed $n \times p$ matrix Y into the sum of a sparse matrix and a low-rank matrix. A convex formulation of the fully-observed problem is

$$\min_{M,S} \|M\|_* + \lambda\|S\|_1 \quad \text{s.t. } Y = M + S,$$

where $\|S\|_1 := \sum_{i=1}^n \sum_{j=1}^p |S_{ij}|$ [11, 14]. Compact, nonconvex formulations that allow noise in the observations include the following:

$$\min_{L,R,S} \frac{1}{2}\|LR^T + S - Y\|_F^2 \quad \text{(fully observed)}$$

$$\min_{L,R,S} \frac{1}{2}\|P_\Phi(LR^T + S - Y)\|_F^2 \quad \text{(partially observed)},$$

where Φ represents the locations of the observed entries of Y and P_Φ is projection onto this set [15, 48].

One application of these formulations is to robust PCA, where the low-rank part represents principal components and the sparse part represents "outlier" observations. Another application is to foreground-background separation in video processing. Here, each column of Y represents the pixels in one frame of video, whereas each row of Y shows the evolution of one pixel over time.

2.8. Subspace Identification In this application, the $a_j \in \mathbb{R}^n$, $j = 1, 2, \ldots, m$ are vectors that lie (approximately) in a low-dimensional subspace. The aim is to identify this subspace, expressed as the column subspace of a matrix $X \in \mathbb{R}^{n\times r}$. If the a_j are fully observed, an obvious way to solve this problem is to perform a singular value decomposition of the $n \times m$ matrix $A = [a_j]_{j=1}^m$, and take X to be the leading r right singular vectors. In interesting variants of this problem, however, the vectors a_j may be arriving in streaming fashion and may be only partly observed, for example in indices $\Phi_j \subset \{1, 2, \ldots, n\}$. We would thus need to identify a matrix X and vectors $s_j \in \mathbb{R}^r$ such that

$$P_{\Phi_j}(a_j - Xs_j) \approx 0, \quad j = 1, 2, \ldots, m.$$

The algorithm for identifying X, described in [1], is a manifold-projection scheme that takes steps in incremental fashion for each a_j in turn. Its validity relies on

incoherence of the matrix X with respect to the principal axes, that is, the matrix X should not have a few elements that are much larger than the others. A local convergence analysis of this method is given in [2].

2.9. Support Vector Machines Classification via support vector machines (SVM) is a classical paradigm in machine learning. This problem takes as input data (a_j, y_j) with $a_j \in \mathbb{R}^n$ and $y_j \in \{-1, 1\}$, and seeks a vector $x \in \mathbb{R}^n$ and a scalar $\beta \in \mathbb{R}$ such that

$$a_j^T x - \beta \geqslant 1 \quad \text{when } y_j = +1; \tag{2.9.1a}$$

$$a_j^T x - \beta \leqslant -1 \quad \text{when } y_j = -1. \tag{2.9.1b}$$

Any pair (x, β) that satisfies these conditions defines a *separating hyperplane* in $\mathbb{R}^n$, that separates the "positive" cases $\{a_j \,|\, y_j = +1\}$ from the "negative" cases $\{a_j \,|\, y_j = -1\}$. (In the language of Section 2.1, we could define the function ϕ as $\phi(a_j) = \text{sign}(a_j^T x - \beta)$.) Among all separating hyperplanes, the one that minimizes $\|x\|^2$ is the one that maximizes the *margin* between the two classes, that is, the hyperplane whose distance to the nearest point a_j of either class is greatest.

We can formulate the problem of finding a separating hyperplane as an optimization problem by defining an objective with the summation form (2.1.2):

$$H(x, \beta) = \frac{1}{m} \sum_{j=1}^{m} \max(1 - y_j(a_j^T x - \beta), 0). \tag{2.9.2}$$

Note that the jth term in this summation is zero if the conditions (2.9.1) are satisfied, and positive otherwise. Even if no pair (x, β) exists with $H(x, \beta) = 0$, the pair (x, β) that minimizes (2.1.2) will be the one that comes as close as possible to satisfying (2.9.1), in a suitable sense. A term $\lambda\|x\|_2^2$, where λ is a small positive parameter, is often added to (2.9.2), yielding the following regularized version:

$$H(x, \beta) = \frac{1}{m} \sum_{j=1}^{m} \max(1 - y_j(a_j^T x - \beta), 0) + \frac{1}{2}\lambda\|x\|_2^2. \tag{2.9.3}$$

If λ is sufficiently small (but positive), and if separating hyperplanes exist, the pair (x, β) that minimizes (2.9.3) is the maximum-margin separating hyperplane. The maximum-margin property is consistent with the goals of generalizability and robustness. For example, if the observed data (a_j, y_j) is drawn from an underlying "cloud" of positive and negative cases, the maximum-margin solution usually does a reasonable job of separating other empirical data samples drawn from the same clouds, whereas a hyperplane that passes close by several of the observed data points may not do as well (see Figure 2.9.4).

The problem of minimizing (2.9.3) can be written as a convex quadratic program — having a convex quadratic objective and linear constraints — by introducing variables s_j, $j = 1, 2, \ldots, m$ to represent the residual terms. Then,

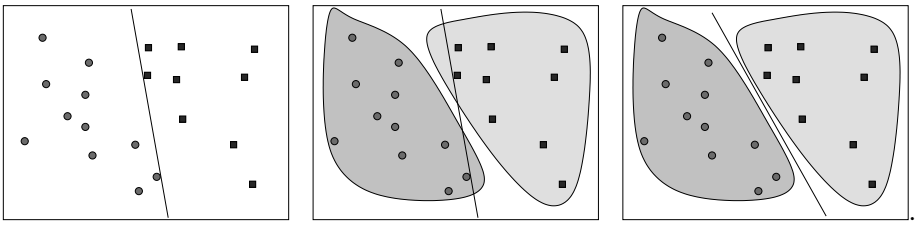

FIGURE 2.9.4. Linear support vector machine classification, with one class represented by circles and the other by squares. One possible choice of separating hyperplane is shown at left. If the observed data is an empirical sample drawn from a cloud of underlying data points, this plane does not do well in separating the two clouds (middle). The maximum-margin separating hyperplane does better (right).

$$\min_{x,\beta,s} \frac{1}{m}\mathbf{1}^T s + \frac{1}{2}\lambda\|x\|_2^2, \tag{2.9.5a}$$

$$\text{subject to} \quad s_j \geqslant 1 - y_j(a_j^T x - \beta), \quad s_j \geqslant 0, \quad j = 1,2,\ldots,m, \tag{2.9.5b}$$

where $\mathbf{1} = (1,1,\ldots,1)^T \in \mathbb{R}^m$.

Often it is not possible to find a hyperplane that separates the positive and negative cases well enough to be useful as a classifier. One solution is to transform all of the raw data vectors a_j by a mapping ζ into a higher-dimensional Euclidean space, then perform the support-vector-machine classification on the vectors $\zeta(a_j)$, $j = 1,2,\ldots,m$.

The conditions (2.9.1) would thus be replaced by

$$\zeta(a_j)^T x - \beta \geqslant 1 \qquad \text{when } y_j = +1; \tag{2.9.6a}$$

$$\zeta(a_j)^T x - \beta \leqslant -1 \qquad \text{when } y_j = -1, \tag{2.9.6b}$$

leading to the following analog of (2.9.3):

$$H(x,\beta) = \frac{1}{m}\sum_{j=1}^{m} \max(1 - y_j(\zeta(a_j)^T x - \beta), 0) + \frac{1}{2}\lambda\|x\|_2^2. \tag{2.9.7}$$

When transformed back to $\mathbb{R}^m$, the surface $\{a \mid \zeta(a)^T x - \beta = 0\}$ is nonlinear and possibly disconnected, and is often a much more powerful classifier than the hyperplanes resulting from (2.9.3).

We can formulate (2.9.7) as a convex quadratic program in exactly the same manner as we derived (2.9.5) from (2.9.3). By taking the dual of this quadratic program, we obtain another convex quadratic program, in m variables:

$$\min_{\alpha\in\mathbb{R}^m} \frac{1}{2}\alpha^T Q\alpha - \mathbf{1}^T\alpha \quad \text{subject to } 0 \leqslant \alpha \leqslant \frac{1}{\lambda}\mathbf{1}, \;\; y^T\alpha = 0, \tag{2.9.8}$$

where

$$Q_{kl} = y_k y_l \zeta(a_k)^T\zeta(a_l), \quad y = (y_1, y_2, \ldots, y_m)^T, \quad \mathbf{1} = (1,1,\ldots,1)^T \in \mathbb{R}^m.$$

Interestingly, problem (2.9.8) can be formulated and solved without any explicit knowledge or definition of the mapping ζ. We need only a technique to define the elements of Q. This can be done with the use of a *kernel function* $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, where $K(a_k, a_l)$ replaces $\zeta(a_k)^T \zeta(a_l)$ [4, 16]. This is the so-called "kernel trick." (The kernel function K can also be used to construct a classification function ϕ from the solution of (2.9.8).) A particularly popular choice of kernel is the Gaussian kernel:

$$K(a_k, a_l) := \exp(-\|a_k - a_l\|^2/(2\sigma)),$$

where σ is a positive parameter.

2.10. Logistic Regression Logistic regression can be viewed as a variant of binary support-vector machine classification, in which rather than the classification function ϕ giving a unqualified prediction of the class in which a new data vector a lies, it returns an estimate of the *odds* of a belonging to one class or the other. We seek an "odds function" p parametrized by a vector $x \in \mathbb{R}^n$ as follows:

$$p(a; x) := (1 + \exp(a^T x))^{-1}, \tag{2.10.1}$$

and aim to choose the parameter x so that

$$p(a_j; x) \approx 1 \quad \text{when } y_j = +1; \tag{2.10.2a}$$

$$p(a_j; x) \approx 0 \quad \text{when } y_j = -1. \tag{2.10.2b}$$

(Note the similarity to (2.9.1).) The optimal value of x can be found by maximizing a log-likelihood function:

$$L(x) := \frac{1}{m}\left[\sum_{j:y_j=-1} \log(1 - p(a_j; x)) + \sum_{j:y_j=1} \log p(a_j; x)\right]. \tag{2.10.3}$$

We can perform feature selection using this model by introducing a regularizer $\lambda\|x\|_1$, as follows:

$$\max_x \frac{1}{m}\left[\sum_{j:y_j=-1} \log(1 - p(a_j; x)) + \sum_{j:y_j=1} \log p(a_j; x)\right] - \lambda\|x\|_1, \tag{2.10.4}$$

where $\lambda > 0$ is a regularization parameter. (Note that we *subtract* rather than add the regularization term $\lambda\|x\|_1$ to the objective, because this problem is formulated as a maximization rather than a minimization.) As we see later, this term has the effect of producing a solution in which few components of x are nonzero, making it possible to evaluate $p(a; x)$ by knowing only those components of a that correspond to the nonzeros in x.

An important extension of this technique is to *multiclass* (or *multinomial*) logistic regression, in which the data vectors a_j belong to more than two classes. Such applications are common in modern data analysis. For example, in a speech recognition system, the M classes could each represent a *phoneme* of speech, one of the potentially thousands of distinct elementary sounds that can be uttered by

humans in a few tens of milliseconds. A multinomial logistic regression problem requires a distinct odds function p_k for each class $k \in \{1, 2, \ldots, M\}$. These functions are parametrized by vectors $x_{[k]} \in \mathbb{R}^n$, $k = 1, 2, \ldots, M$, defined as follows:

$$p_k(a; X) := \frac{\exp(a^T x_{[k]})}{\sum_{l=1}^{M} \exp(a^T x_{[l]})}, \quad k = 1, 2, \ldots, M, \tag{2.10.5}$$

where we define $X := \{x_{[k]} \,|\, k = 1, 2, \ldots, M\}$. Note that for all a and for all $k = 1, 2, \ldots, M$, we have $p_k(a) \in (0, 1)$ and also $\sum_{k=1}^{M} p_k(a) = 1$. The operation in (2.10.5) is referred to as a "softmax" on the quantities $\{a^T x_{[l]} \,|\, l = 1, 2, \ldots, M\}$. If one of these inner products dominates the others, that is, $a^T x_{[k]} \gg a^T x_{[l]}$ for all $l \neq k$, the formula (2.10.5) will yield $p_k(a; X) \approx 1$ and $p_l(a; X) \approx 0$ for all $l \neq k$.

In the setting of multiclass logistic regression, the labels y_j are vectors in $\mathbb{R}^M$, whose elements are defined as follows:

$$y_{jk} = \begin{cases} 1 & \text{when } a_j \text{ belongs to class } k, \\ 0 & \text{otherwise.} \end{cases} \tag{2.10.6}$$

Similarly to (2.10.2), we seek to define the vectors $x_{[k]}$ so that

$$p_k(a_j; X) \approx 1 \quad \text{when } y_{jk} = 1 \tag{2.10.7a}$$

$$p_k(a_j; X) \approx 0 \quad \text{when } y_{jk} = 0. \tag{2.10.7b}$$

The problem of finding values of $x_{[k]}$ that satisfy these conditions can again be formulated as one of maximizing a log-likelihood:

$$L(X) := \frac{1}{m} \sum_{j=1}^{m} \left[\sum_{\ell=1}^{M} y_{j\ell} (x_{[\ell]}^T a_j) - \log \left(\sum_{\ell=1}^{M} \exp(x_{[\ell]}^T a_j) \right) \right]. \tag{2.10.8}$$

"Group-sparse" regularization terms can be included in this formulation to select a set of features in the vectors a_j, common to each class, that distinguish effectively between the classes.

2.11. Deep Learning Deep neural networks are often designed to perform the same function as multiclass logistic regression, that is, to classify a data vector a into one of M possible classes, where $M \geqslant 2$ is large in some key applications. The difference is that the data vector a undergoes a series of structured transformations before being passed through a multiclass logistic regression classifier of the type described in the previous subsection.

The simple neural network shown in Figure 2.11.1 illustrates the basic ideas. In this figure, the data vector a_j enters at the bottom of the network, each node in the bottom layer corresponding to one component of a_j. The vector then moves upward through the network, undergoing a structured nonlinear transformation as it moves from one layer to the next. A typical form of this transformation, which converts the vector a_j^{l-1} at layer $l-1$ to input vector a_j^l at layer l, is

$$a_j^l = \sigma(W^l a_j^{l-1} + g^l), \quad l = 1, 2, \ldots, D,$$

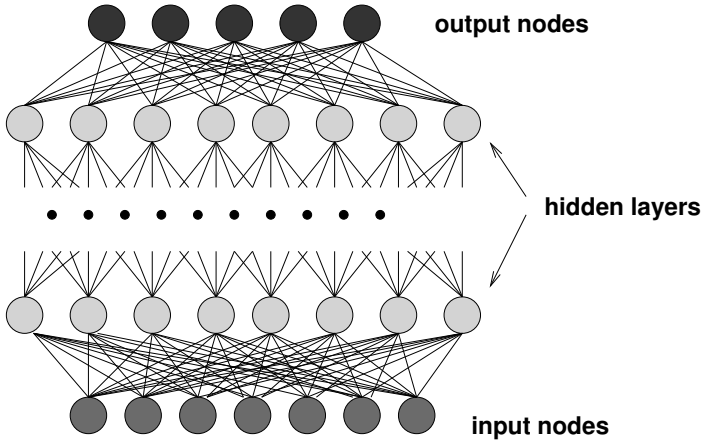

FIGURE 2.11.1. A deep neural network, showing connections between adjacent layers.

where W^l is a matrix of dimension $|a_j^l| \times |a_j^{l-1}|$ and g^l is a vector of length $|a_j^l|$, σ is a *componentwise* nonlinear transformation, and D is the number of *hidden layers*, defined as the layers situated strictly between the bottom and top layers. Each arc in Figure 2.11.1 represents one of the elements of a transformation matrix W^l. We define a_j^0 to be the "raw" input vector a_j, and let a_j^D be the vector formed by the nodes at the topmost hidden layer in Figure 2.11.1. Typical forms of the function σ include the following, acting identically on each component $t \in \mathbb{R}$ of its input vector:

- Logistic function: $t \to 1/(1+e^{-t})$;
- Hinge loss: $t \to \max(t, 0)$;
- Bernoulli: a random function that outputs 1 with probability $1/(1+e^{-t})$ and 0 otherwise.

Each node in the top layer corresponds to a particular class, and the output of each node corresponds to the odds of the input vector belonging to each class. As mentioned, the "softmax" operator is typically used to convert the transformed input vector in the second-top layer (layer D) to a set of odds at the top layer. Associated with each input vector a_j are labels y_{jk}, defined as in (2.10.6) to indicate which of the M classes that a_j belongs to.

The parameters in this neural network are the matrix-vector pairs (W^l, g^l), $l = 1, 2, \ldots, D$ that transform the input vector a_j into its form a_j^D at the topmost hidden layer, together with the parameters X of the multiclass logistic regression operation that takes place at the very top stage, where X is defined exactly as in the discussion of Section 2.10. We aim to choose all these parameters so that the network does a good job on classifying the training data correctly. Using the notation w for the hidden layer transformations, that is,

$$w := (W^1, g^1, W^2, g^2, \ldots, W^D, g^D), \tag{2.11.2}$$

and defining $X := \{x_{[k]} \mid k = 1, 2, \ldots, M\}$ as in Section 2.10, we can write the loss function for deep learning as follows:

$$L(w, X) := \frac{1}{m} \sum_{j=1}^{m} \left[\sum_{\ell=1}^{M} y_{j\ell} (x_{[\ell]}^T a_j^D(w)) - \log \left(\sum_{\ell=1}^{M} \exp(x_{[\ell]}^T a_j^D(w)) \right) \right]. \tag{2.11.3}$$

Note that this is exactly the function (2.10.8) applied to the output of the top hidden layer $a_j^D(w)$. We write $a_j^D(w)$ to make explicit the dependence of a_j^D on the parameters w of (2.11.2), as well as on the input vector a_j. (We can view multiclass logistic regression (2.10.8) as a special case of deep learning in which there are no hidden layers, so that $D = 0$, w is null, and $a_j^D = a_j$, $j = 1, 2, \ldots, m$.)

Neural networks in use for particular applications (in image recognition and speech recognition, for example, where they have been very successful) include many variants on the basic design above. These include restricted connectivity between layers (that is, enforcing structure on the matrices W^l, $l = 1, 2, \ldots, D$), layer arrangements that are more complex than the linear layout illustrated in Figure 2.11.1, with outputs coming from different levels, connections across non-adjacent layers, different componentwise transformations σ at different layers, and so on. Deep neural networks for practical applications are highly engineered objects.

The loss function (2.11.3) shares with many other applications the "summation" form (2.1.2), but it has several features that set it apart from the other applications discussed above. First, and possibly most important, it is *nonconvex* in the parameters w. There is reason to believe that the "landscape" of L is complex, with the global minimizer being exceedingly difficult to find. Second, the total number of parameters in (w, X) is usually very large. The most popular algorithms for minimizing (2.11.3) are of stochastic gradient type, which like most optimization methods come with no guarantee for finding the minimizer of a nonconvex function. Effective training of deep learning classifiers typically requires a great deal of data and computation power. Huge clusters of powerful computers, often using multicore processors, GPUs, and even specially architected processing units, are devoted to this task. Efficiency also requires many heuristics in the formulation and the algorithm (for example, in the choice of regularization functions and in the steplengths for stochastic gradient).

3. Preliminaries

We discuss here some foundations for the analysis of subsequent sections. These include useful facts about smooth and nonsmooth convex functions, Taylor's theorem and some of its consequences, optimality conditions, and proximal operators.

In the discussion of this section, our basic assumption is that f is a mapping from $\mathbb{R}^n$ to $\mathbb{R} \cup \{+\infty\}$, continuous on its effective domain $D := \{x \mid f(x) < \infty\}$. Further assumptions of f are introduced as needed.

3.1. Solutions Consider the problem of minimizing f (1.0.1). We have the following terminology:

- x^* is a *local minimizer* of f if there is a neighborhood N of x^* such that $f(x) \geqslant f(x^*)$ for all $x \in N$.
- x^* is a *global minimizer* of f if $f(x) \geqslant f(x^*)$ for all $x \in \mathbb{R}^n$.
- x^* is a *strict local minimizer* if it is a local minimizer on some neighborhood N and in addition $f(x) > f(x^*)$ for all $x \in N$ with $x \neq x^*$.
- x^* is an *isolated local minimizer* if there is a neighborhood N of x^* such that $f(x) \geqslant f(x^*)$ for all $x \in N$ and in addition, N contains no local minimizers other than x^*.

3.2. Convexity and Subgradients A convex set $\Omega \subset \mathbb{R}^n$ has the property that

$$x, y \in \Omega \quad \Rightarrow \quad (1-\alpha)x + \alpha y \in \Omega \ \text{ for all } \alpha \in [0,1]. \tag{3.2.1}$$

We usually deal with *closed* convex sets in this article. For a convex set $\Omega \subset \mathbb{R}^n$ we define the *indicator function* $I_\Omega(x)$ as follows:

$$I_\Omega(x) = \begin{cases} 0 & \text{if } x \in \Omega \\ +\infty & \text{otherwise.} \end{cases}$$

Indicator functions are useful devices for deriving optimality conditions for constrained problems, and even for developing algorithms. The constrained optimization problem

$$\min_{x \in \Omega} \ f(x) \tag{3.2.2}$$

can be restated equivalently as follows:

$$\min \ f(x) + I_\Omega(x). \tag{3.2.3}$$

We noted already that a convex function $\phi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ has the following defining property:

$$\phi((1-\alpha)x + \alpha y) \leqslant (1-\alpha)\phi(x) + \alpha\phi(y), \quad \text{for all } x, y \in \mathbb{R}^n \text{ and all } \alpha \in [0,1]. \tag{3.2.4}$$

The concepts of "minimizer" are simpler in the case of convex objective functions than in the general case. In particular, the distinction between "local" and "global" minimizers disappears. For f convex in (1.0.1), we have the following.

(a) Any local minimizer of (1.0.1) is also a global minimizer.
(b) The set of global minimizers of (1.0.1) is a convex set.

If there exists a value $\gamma > 0$ such that

$$\phi((1-\alpha)x + \alpha y) \leqslant (1-\alpha)\phi(x) + \alpha\phi(y) - \frac{1}{2}\gamma\alpha(1-\alpha)\|x-y\|_2^2 \tag{3.2.5}$$

for all x and y in the domain of ϕ and $\alpha \in [0,1]$, we say that ϕ is *strongly convex with modulus of convexity* γ.

We summarize some definitions and results about subgradients of convex functions here. For a more extensive discussion, see [22].

Definition 3.2.6. A vector $v \in \mathbb{R}^n$ is a *subgradient* of f at a point x if

$$f(x+d) \geqslant f(x) + v^T d. \quad \text{for all } d \in \mathbb{R}^n.$$

The *subdifferential*, denoted $\partial f(x)$, is the set of all subgradients of f at x.

Subdifferentials satisfy a *monotonicity* property, as we show now.

Lemma 3.2.7. *If $a \in \partial f(x)$ and $b \in \partial f(y)$, we have $(a-b)^T(x-y) \geqslant 0$.*

Proof. From the convexity of f and the definitions of a and b, we deduce that $f(y) \geqslant f(x) + a^T(y-x)$ and $f(x) \geqslant f(y) + b^T(x-y)$. The result follows by adding these two inequalities. □

We can easily characterize a minimum in terms of the subdifferential.

Theorem 3.2.8. *The point x^* is the minimizer of a convex function f if and only if $0 \in \partial f(x^*)$.*

Proof. Suppose that $0 \in \partial f(x^*)$, we have by substituting $x = x^*$ and $v = 0$ into Definition 3.2.6 that $f(x^* + d) \geqslant f(x^*)$ for all $d \in \mathbb{R}^n$, which implies that x^* is a minimizer of f.

The converse follows trivially by showing that $v = 0$ satisfies Definition 3.2.6 when x^* is a minimizer. □

The subdifferential is the generalization to nonsmooth convex functions of the concept of derivative of a smooth function.

Theorem 3.2.9. *If f is convex and differentiable at x, then $\partial f(x) = \{\nabla f(x)\}$.*

A converse of this result is also true. Specifically, if the subdifferential of a convex function f at x contains a single subgradient, then f is differentiable with gradient equal to this subgradient (see [40, Theorem 25.1]).

3.3. Taylor's Theorem Taylor's theorem is a foundational result for optimization of smooth nonlinear functions. It shows how smooth functions can be approximated locally by low-order (linear or quadratic) functions.

Theorem 3.3.1. *Given a continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, and given $x, p \in \mathbb{R}^n$, we have that*

$$f(x+p) = f(x) + \int_0^1 \nabla f(x+\xi p)^T p \, d\xi, \tag{3.3.2}$$

$$f(x+p) = f(x) + \nabla f(x+\xi p)^T p, \quad \textit{some } \xi \in (0,1). \tag{3.3.3}$$

If f is twice continuously differentiable, we have

$$\nabla f(x+p) = \nabla f(x) + \int_0^1 \nabla^2 f(x+\xi p) p \, d\xi, \tag{3.3.4}$$

$$f(x+p) = f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x+\xi p) p, \quad \textit{for some } \xi \in (0,1). \tag{3.3.5}$$

We can derive an important consequence of this theorem when f is *Lipschitz* continuously differentiable with constant L, that is,

$$\|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\|, \quad \text{for all } x, y \in \mathbb{R}^n. \tag{3.3.6}$$

We have by setting $y = x + p$ in (3.3.2) and subtracting the term $\nabla f(x)^T(y - x)$ from both sides that

$$f(y) - f(x) - \nabla f(x)^T(y - x) = \int_0^1 [\nabla f(x + \xi(y - x)) - \nabla f(x)]^T(y - x)\, d\xi.$$

By using (3.3.6), we have

$$\begin{aligned}[\nabla f(x + \xi(y - x)) - \nabla f(x)]^T(y - x) &\leqslant \|\nabla f(x + \xi(y - x)) - \nabla f(x)\|\|y - x\| \\ &\leqslant L\xi\|y - x\|^2.\end{aligned}$$

By substituting this bound into the previous integral, we obtain

$$f(y) - f(x) - \nabla f(x)^T(y - x) \leqslant \frac{L}{2}\|y - x\|^2. \tag{3.3.7}$$

For the remainder of Section 3.3, we assume that f is continuously differentiable and also *convex*. The definition of convexity (3.2.4) and the fact that $\partial f(x) = \{\nabla f(x)\}$ implies that

$$f(y) \geqslant f(x) + \nabla f(x)^T(y - x), \quad \text{for all } x, y \in \mathbb{R}^n. \tag{3.3.8}$$

We defined "strong convexity with modulus γ" in (3.2.5). When f is differentiable, we have the following equivalent definition, obtained by rearranging (3.2.5) and letting $\alpha \downarrow 0$.

$$f(y) \geqslant f(x) + \nabla f(x)^T(y - x) + \frac{\gamma}{2}\|y - x\|^2. \tag{3.3.9}$$

By combining this expression with (3.3.7), we have the following result.

Lemma 3.3.10. *Given convex f satisfying (3.2.5), with ∇f uniformly Lipschitz continuous with constant L, we have for any x, y that*

$$\frac{\gamma}{2}\|y - x\|^2 \leqslant f(y) - f(x) - \nabla f(x)^T(y - x) \leqslant \frac{L}{2}\|y - x\|^2. \tag{3.3.11}$$

For later convenience, we define a *condition number* κ as follows:

$$\kappa := \frac{L}{\gamma}. \tag{3.3.12}$$

When f is *twice* continuously differentiable, we can characterize the constants γ and L in terms of the eigenvalues of the Hessian $\nabla f(x)$. Specifically, we can show that (3.3.11) is equivalent to

$$\gamma I \preceq \nabla^2 f(x) \preceq LI, \quad \text{for all } x. \tag{3.3.13}$$

When f is strictly convex and quadratic, κ defined in (3.3.12) is the condition number of the (constant) Hessian, in the usual sense of linear algebra.

Strongly convex functions have unique minimizers, as we now show.

Theorem 3.3.14. *Let f be differentiable and strongly convex with modulus $\gamma > 0$. Then the minimizer x^* of f exists and is unique.*

Proof. We show first that for any point x^0, the level set $\{x \mid f(x) \leqslant f(x^0)\}$ is closed and bounded, and hence compact. Suppose for contradiction that there is a sequence $\{x^\ell\}$ such that $\|x^\ell\| \to \infty$ and

$$f(x^\ell) \leqslant f(x^0). \tag{3.3.15}$$

By strong convexity of f, we have for some $\gamma > 0$ that

$$f(x^\ell) \geqslant f(x^0) + \nabla f(x^0)^T(x^\ell - x^0) + \frac{\gamma}{2}\|x^\ell - x^0\|^2.$$

By rearranging slightly, and using (3.3.15), we obtain

$$\frac{\gamma}{2}\|x^\ell - x^0\|^2 \leqslant -\nabla f(x^0)^T(x^\ell - x^0) \leqslant \|\nabla f(x^0)\|\|x^\ell - x^0\|.$$

By dividing both sides by $(\gamma/2)\|x^\ell - x^0\|$, we obtain $\|x^\ell - x^0\| \leqslant (2/\gamma)\|\nabla f(x^0)\|$ for all ℓ, which contradicts unboundedness of $\{x^\ell\}$. Thus, the level set is bounded. Since it is also closed (by continuity of f), it is compact.

Since f is continuous, it attains its minimum on the compact level set, which is also the solution of $\min_x f(x)$, and we denote it by x^*. Suppose for contradiction that the minimizer is not unique, so that we have two points x_1^* and x_2^* that minimize f. Obviously, these points must attain equal objective values, so that $f(x_1^*) = f(x_2^*) = f^*$ for some f^*. By taking (3.2.5) and setting $\phi = f^*$, $x = x_1^*$, $y = x_2^*$, and $\alpha = 1/2$, we obtain

$$f((x_1^* + x_2^*)/2) \leqslant \frac{1}{2}(f(x_1^*) + f(x_2^*)) - \frac{1}{8}\gamma\|x_1^* - x_2^*\|^2 < f^*,$$

so the point $(x_1^* + x_2^*)/2$ has a smaller function value than both x_1^* and x_2^*, contradicting our assumption that x_1^* and x_2^* are both minimizers. Hence, the minimizer x^* is unique. □

3.4. Optimality Conditions for Smooth Functions We consider the case of a smooth (twice continuously differentiable) function f that is not necessarily convex. Before designing algorithms to find a minimizer of f, we need to identify properties of f and its derivatives at a point $\bar{x}$ that tell us whether or not $\bar{x}$ is a minimizer, of one of the types described in Subsection 3.1. We call such properties *optimality conditions*.

A *first-order necessary* condition for optimality is that $\nabla f(\bar{x}) = 0$. More precisely, if $\bar{x}$ is a local minimizer, then $\nabla f(\bar{x}) = 0$. We can prove this by using Taylor's theorem. Supposing for contradiction that $\nabla f(\bar{x}) \neq 0$, we can show by setting $x = \bar{x}$ and $p = -\alpha\nabla f(\bar{x})$ for $\alpha > 0$ in (3.3.3) that $f(\bar{x} - \alpha\nabla f(\bar{x})) < f(\bar{x})$ for all $\alpha > 0$ sufficiently small. Thus *any* neighborhood of $\bar{x}$ will contain points x with a $f(x) < f(\bar{x})$, so $\bar{x}$ cannot be a local minimizer.

If f is convex, as well as smooth, the condition $\nabla f(\bar{x}) = 0$ is *sufficient* for $\bar{x}$ to be a *global* solution. This claim follows immediately from Theorems 3.2.8 and 3.2.9.

A *second-order necessary condition* for $\bar{x}$ to be a local solution is that $\nabla f(\bar{x}) = 0$ and $\nabla^2 f(\bar{x})$ is positive semidefinite. The proof is by an argument similar to that of the first-order necessary condition, but using the second-order Taylor series expansion (3.3.5) instead of (3.3.3). A *second-order sufficient condition* is that $\nabla f(\bar{x}) = 0$

and $\nabla^2 f(\bar{x})$ is positive definite. This condition guarantees that $\bar{x}$ is a *strict* local minimizer, that is, there is a neighborhood of $\bar{x}$ such that $\bar{x}$ has a strictly smaller function value than all other points in this neighborhood. Again, the proof makes use of (3.3.5).

We call $\bar{x}$ a *stationary point* for smooth f if it satisfies the first-order necessary condition $\nabla f(\bar{x}) = 0$. Stationary points are not necessarily local minimizers. In fact, local *maximizers* satisfy the same condition. More interestingly, stationary points can be *saddle points*. These are points for which there exist directions u and v such that $f(\bar{x} + \alpha u) < f(\bar{x})$ and $f(\bar{x} + \alpha v) > f(\bar{x})$ for all positive α sufficiently small. When the Hessian $\nabla^2 f(\bar{x})$ has both strictly positive and strictly negative eigenvalues, it follows from (3.3.5) that $\bar{x}$ is a saddle point. When $\nabla^2 f(\bar{x})$ is positive semidefinite or negative semidefinite, second derivatives alone are insufficient to classify $\bar{x}$; higher-order derivative information is needed.

3.5. Proximal Operators and the Moreau Envelope Here we present some analysis for analyzing the convergence of algorithms for the regularized problem (1.0.2), where the objective is the sum of a smooth function and a convex (usually nonsmooth) function.

We start with a formal definition.

Definition 3.5.1. For a closed proper convex function h and a positive scalar λ, the *Moreau envelope* is

$$M_{\lambda,h}(x) := \inf_u \left\{ h(u) + \frac{1}{2\lambda}\|u - x\|^2 \right\} = \frac{1}{\lambda} \inf_u \left\{ \lambda h(u) + \frac{1}{2}\|u - x\|^2 \right\}. \tag{3.5.2}$$

The proximal operator of the function λh is the value of u that achieves the infimum in (3.5.2), that is,

$$\operatorname{prox}_{\lambda h}(x) := \arg\min_u \left\{ \lambda h(u) + \frac{1}{2}\|u - x\|^2 \right\}. \tag{3.5.3}$$

From optimality properties for (3.5.3) (see Theorem 3.2.8), we have

$$0 \in \lambda \partial h(\operatorname{prox}_{\lambda h}(x)) + (\operatorname{prox}_{\lambda h}(x) - x). \tag{3.5.4}$$

The Moreau envelope can be viewed as a kind of smoothing or regularization of the function h. It has a finite value for all x, even when h takes on infinite values for some $x \in \mathbb{R}^n$. In fact, it is differentiable everywhere, with gradient

$$\nabla M_{\lambda,h}(x) = \frac{1}{\lambda}(x - \operatorname{prox}_{\lambda h}(x)).$$

Moreover, x^* is a minimizer of h if and only if it is a minimizer of $M_{\lambda,h}$.

The proximal operator satisfies a nonexpansiveness property. From the optimality conditions (3.5.4) at two points x and y, we have

$$x - \operatorname{prox}_{\lambda h}(x) \in \lambda \partial(\operatorname{prox}_{\lambda h}(x)), \quad y - \operatorname{prox}_{\lambda h}(y) \in \lambda \partial(\operatorname{prox}_{\lambda h}(y)).$$

By applying monotonicity (Lemma 3.2.7), we have

$$(1/\lambda)\big((x - \operatorname{prox}_{\lambda h}(x)) - (y - \operatorname{prox}_{\lambda h}(y))\big)^T (\operatorname{prox}_{\lambda h}(x) - \operatorname{prox}_{\lambda h}(y)) \geqslant 0,$$

Rearranging this and applying the Cauchy-Schwartz inequality yields

$$\begin{aligned}\|\text{prox}_{\lambda h}(x) - \text{prox}_{\lambda h}(y)\|^2 &\leqslant (x-y)^T(\text{prox}_{\lambda h}(x) - \text{prox}_{\lambda h}(y)) \\ &\leqslant \|x-y\|\,\|\text{prox}_{\lambda h}(x) - \text{prox}_{\lambda h}(y)\|,\end{aligned}$$

from which we obtain $\|\text{prox}_{\lambda h}(x) - \text{prox}_{\lambda h}(y)\| \leqslant \|x-y\|$, as claimed.

We list the prox operator for several instances of h that are common in data analysis applications. These definitions are useful in implementing the prox-gradient algorithms of Section 5.

- $h(x) = 0$ for all x, for which we have $\text{prox}_{\lambda h}(x) = 0$. (This observation is useful in proving that the prox-gradient method reduces to the familiar steepest descent method when the objective contains no regularization term.)
- $h(x) = I_\Omega(x)$, the indicator function for a closed convex set Ω. In this case, we have for any $\lambda > 0$ that

$$\text{prox}_{\lambda I_\Omega}(x) = \arg\min_u \left\{\lambda I_\Omega(u) + \frac{1}{2}\|u-x\|^2\right\} = \arg\min_{u\in\Omega} \frac{1}{2}\|u-x\|^2,$$

  which is simply the projection of x onto the set Ω.
- $h(x) = \|x\|_1$. By substituting into definition (3.5.3) we see that the minimization separates into its n separate components, and that the ith component of $\text{prox}_{\lambda\|\cdot\|_1}(x)$ is

$$\left[\text{prox}_{\lambda\|\cdot\|_1}(x)\right]_i = \arg\min_{u_i} \left\{\lambda|u_i| + \frac{1}{2}(u_i - x_i)^2\right\}.$$

  We can thus verify that

$$[\text{prox}_{\lambda\|\cdot\|_1}(x)]_i = \begin{cases} x_i - \lambda & \text{if } x_i > \lambda; \\ 0 & \text{if } x_i \in [-\lambda, \lambda]; \\ x_i + \lambda & \text{if } x_i < -\lambda, \end{cases} \tag{3.5.5}$$

  an operation that is known as *soft-thresholding*.
- $h(x) = \|x\|_0$, where $\|x\|_0$ denotes the *cardinality* of the vector x, its number of nonzero components. Although this h is not a convex function (as we can see by considering convex combinations of the vectors $(0,1)^T$ and $(1,0)^T$ in $\mathbb{R}^2$), its proximal operator is well defined, and is known as *hard thresholding*:

$$[\text{prox}_{\lambda\|\cdot\|_0}(x)]_i = \begin{cases} x_i & \text{if } |x_i| \geqslant \sqrt{2\lambda}; \\ 0 & \text{if } |x_i| < \sqrt{2\lambda}. \end{cases}$$

  As in (3.5.5), the definition (3.5.3) separates into n individual components.

3.6. Convergence Rates An important measure for evaluating algorithms is the rate of convergence to zero of some measure of error. For smooth f, we may be interested in how rapidly the sequence of gradient norms $\{\|\nabla f(x^k)\|\}$ converges to zero. For nonsmooth convex f, a measure of interest may be convergence to

zero of $\{\text{dist}(0, \partial f(x^k))\}$ (the sequence of distances from 0 to the subdifferential $\partial f(x^k)$). Other error measures for which we may be able to prove convergence rates include $\|x^k - x^*\|$ (where x^* is a solution) and $f(x^k) - f^*$ (where f^* is the optimal value of the objective function f). For generality, we denote by $\{\phi_k\}$ the sequence of nonnegative scalars whose rate of convergence to 0 we wish to find.

We say that *linear* convergence holds if there is some $\sigma \in (0,1)$ such that

$$\phi_{k+1}/\phi_k \leqslant 1 - \sigma, \quad \text{for all } k \text{ sufficiently large.} \tag{3.6.1}$$

(This property is sometimes also called *geometric* or *exponential* convergence, but the term *linear* is standard in the optimization literature, so we use it here.) It follows from (3.6.1) that there is some positive constant C such that

$$\phi_k \leqslant C(1-\sigma)^k, \quad k = 1, 2, \ldots . \tag{3.6.2}$$

While (3.6.1) implies (3.6.2), the converse does not hold. The sequence

$$\phi_k = \begin{cases} 2^{-k} & k \text{ even} \\ 0 & k \text{ odd}, \end{cases}$$

satisfies (3.6.2) with $C = 1$ and $\sigma = .5$, but does not satisfy (3.6.1). To distinguish between these two slightly different definitions, (3.6.1) is sometimes called *Q-linear* while (3.6.2) is called *R-linear*.

Sublinear convergence is, as its name suggests, slower than linear. Several varieties of sublinear convergence are encountered in optimization algorithms for data analysis, including the following

$$\phi_k \leqslant C/\sqrt{k}, \quad k = 1, 2, \ldots, \tag{3.6.3a}$$

$$\phi_k \leqslant C/k, \quad k = 1, 2, \ldots, \tag{3.6.3b}$$

$$\phi_k \leqslant C/k^2, \quad k = 1, 2, \ldots, \tag{3.6.3c}$$

where in each case, C is some positive constant.

Superlinear convergence occurs when the constant $\sigma \in (0,1)$ in (3.6.1) can be chosen arbitrarily close to 1. Specifically, we say that the sequence $\{\phi_k\}$ converges *Q-superlinearly* to 0 if

$$\lim_{k \to \infty} \phi_{k+1}/\phi_k = 0. \tag{3.6.4}$$

Q-Quadratic convergence occurs when

$$\phi_{k+1}/\phi_k^2 \leqslant C, \quad k = 1, 2, \ldots, \tag{3.6.5}$$

for some sufficiently large C. We say that the convergence is *R-superlinear* if there is a Q-superlinearly convergent sequence $\{\nu_k\}$ that dominates $\{\phi_k\}$ (that is, $0 \leqslant \phi_k \leqslant \nu_k$ for all k). R-quadratic convergence is defined similarly. Quadratic and superlinear rates are associated with higher-order methods, such as Newton and quasi-Newton methods.

When a convergence rate applies *globally*, from any reasonable starting point, it can be used to derive a complexity bound for the algorithm, which takes the

form of a bound on the number of iterations K required to reduce ϕ_k below some specified tolerance ϵ. For a sequence satisfying the R-linear convergence condition (3.6.2) a sufficient condition for $\phi_K \leqslant \epsilon$ is $C(1-\sigma)^K \leqslant \epsilon$. By using the estimate $\log(1-\sigma) \leqslant -\sigma$ for all $\sigma \in (0,1)$, we have that

$$C(1-\sigma)^K \leqslant \epsilon \quad \Leftrightarrow \quad K\log(1-\sigma) \leqslant \log(\epsilon/C) \quad \Leftarrow \quad K \geqslant \log(C/\epsilon)/\sigma.$$

It follows that for linearly convergent algorithms, the number of iterations required to converge to a tolerance ϵ depends logarithmically on $1/\epsilon$ and inversely on the rate constant σ. For an algorithm that satisfies the sublinear rate (3.6.3a), a sufficient condition for $\phi_K \leqslant \epsilon$ is $C/\sqrt{K} \leqslant \epsilon$, which is equivalent to $K \geqslant (C/\epsilon)^2$, so the complexity is $O(1/\epsilon^2)$. Similar analyses for (3.6.3b) reveal complexity of $O(1/\epsilon)$, while for (3.6.3c), we have complexity $O(1/\sqrt{\epsilon})$.

For quadratically convergent methods, the complexity is doubly logarithmic in ϵ (that is, $O(\log\log(1/\epsilon))$). Once the algorithm enters a neighborhood of quadratic convergence, just a few additional iterations are required for convergence to a solution of high accuracy.

4. Gradient Methods

We consider here iterative methods for solving the unconstrained smooth problem (1.0.1) that make use of the gradient ∇f (see also, [22] which describes subgradient methods for nonsmooth convex functions.) We consider mostly methods that generate an iteration sequence $\{x^k\}$ via the formula

$$x^{k+1} = x^k + \alpha_k d^k, \tag{4.0.1}$$

where d^k is the search direction and α_k is a steplength.

We consider the steepest descent method, which searches along the negative gradient direction $d^k = -\nabla f(x^k)$, proving convergence results for nonconvex functions, convex functions, and strongly convex functions. In Subsection 4.5, we consider methods that use more general descent directions d^k, proving convergence of methods that make careful choices of the line search parameter α_k at each iteration. In Subsection 4.6, we consider the conditional gradient method for minimization of a smooth function f over a compact set.

4.1. Steepest Descent The simplest stepsize protocol is the short-step variant of steepest descent. We assume here that f is differentiable, with gradient ∇f satisfying the Lipschitz continuity condition (3.3.6) with constant L. We choose the search direction $d^k = -\nabla f(x^k)$ in (4.0.1), and set the steplength α_k to be the constant $1/L$, to obtain the iteration

$$x^{k+1} = x^k - \frac{1}{L}\nabla f(x^k), \quad k = 0, 1, 2, \ldots . \tag{4.1.1}$$

To estimate the amount of decrease in f obtained at each iterate of this method, we use Taylor's theorem. From (3.3.7), we have

$$f(x+\alpha d) \leqslant f(x) + \alpha \nabla f(x)^T d + \alpha^2 \frac{L}{2}\|d\|^2, \tag{4.1.2}$$

For $x = x^k$ and $d = -\nabla f(x^k)$, the value of α that minimizes the expression on the right-hand side is $\alpha = 1/L$. By substituting these values, we obtain

$$f(x^{k+1}) = f(x^k - (1/L)\nabla f(x^k)) \leqslant f(x^k) - \frac{1}{2L}\|\nabla f(x^k)\|^2. \tag{4.1.3}$$

This expression is one of the foundational inequalities in the analysis of optimization methods. Depending on the assumptions about f, we can derive a variety of different convergence rates from this basic inequality.

4.2. General Case We consider first a function f that is Lipschitz continuously differentiable and bounded below, but that need not necessarily be convex. Using (4.1.3) alone, we can prove a sublinear convergence result for the steepest descent method.

Theorem 4.2.1. *Suppose that f is Lipschitz continuously differentiable, satisfying (3.3.6), and that f is bounded below by a constant $\bar{f}$. Then for the steepest descent method with constant steplength $\alpha_k \equiv 1/L$, applied from a starting point x^0, we have for any integer $T \geqslant 1$ that*

$$\min_{0\leqslant k\leqslant T-1} \|\nabla f(x^k)\| \leqslant \sqrt{\frac{2L[f(x^0) - f(x^T)]}{T}} \leqslant \sqrt{\frac{2L[f(x^0) - \bar{f}]}{T}}.$$

Proof. Rearranging (4.1.3) and summing over the first $T-1$ iterates, we have

$$\sum_{k=0}^{T-1} \|\nabla f(x^k)\|^2 \leqslant 2L \sum_{k=0}^{T-1} [f(x^k) - f(x^{k+1})] = 2L[f(x^0) - f(x^T)]. \tag{4.2.2}$$

(Note the telescoping sum.) Since f is bounded below by $\bar{f}$, the right-hand side is bounded above by the constant $2L[f(x^0) - \bar{f}]$. We also have that

$$\min_{0\leqslant k\leqslant T-1} \|\nabla f(x^k)\| = \sqrt{\min_{0\leqslant k\leqslant T-1} \|\nabla f(x^k)\|^2} \leqslant \sqrt{\frac{1}{T}\sum_{k=0}^{T-1} \|\nabla f(x^k)\|^2}.$$

The result is obtained by combining this bound with (4.2.2). □

This result shows that within the first $T-1$ steps of steepest descent, at least one of the iterates has gradient norm less than $\sqrt{2L[f(x^0) - \bar{f}]/T}$, which represents sublinear convergence of type (3.6.3a). It follows too from (4.2.2) that for f bounded below, any accumulation point of the sequence $\{x^k\}$ is stationary.

4.3. Convex Case When f is also convex, we have the following stronger result for the steepest descent method.

Theorem 4.3.1. *Suppose that f is convex and Lipschitz continuously differentiable, satisfying (3.3.6), and that (1.0.1) has a solution x^*. Then the steepest descent method with stepsize $\alpha_k \equiv 1/L$ generates a sequence $\{x^k\}_{k=0}^{\infty}$ that satisfies*

$$f(x^T) - f^* \leqslant \frac{L}{2T}\|x^0 - x^*\|^2. \tag{4.3.2}$$

Proof. By convexity of f, we have $f(x^*) \geqslant f(x^k) + \nabla f(x^k)^T(x^* - x^k)$, so by substituting into (4.1.3), we obtain for $k = 0, 1, 2, \ldots$ that

$$\begin{aligned} f(x^{k+1}) &\leqslant f(x^*) + \nabla f(x^k)^T(x^k - x^*) - \frac{1}{2L}\|\nabla f(x^k)\|^2 \\ &= f(x^*) + \frac{L}{2}\left(\|x^k - x^*\|^2 - \left\|x^k - x^* - \frac{1}{L}\nabla f(x^k)\right\|^2\right) \\ &= f(x^*) + \frac{L}{2}\left(\|x^k - x^*\|^2 - \|x^{k+1} - x^*\|^2\right). \end{aligned}$$

By summing over $k = 0, 1, 2, \ldots, T-1$, and noting the telescoping sum, we have

$$\begin{aligned} \sum_{k=0}^{T-1}(f(x^{k+1}) - f^*) &\leqslant \frac{L}{2}\sum_{k=0}^{T-1}\left(\|x^k - x^*\|^2 - \|x^{k+1} - x^*\|^2\right) \\ &= \frac{L}{2}\left(\|x^0 - x^*\|^2 - \|x^T - x^*\|^2\right) \\ &\leqslant \frac{L}{2}\|x^0 - x^*\|^2. \end{aligned}$$

Since $\{f(x^k)\}$ is a nonincreasing sequence, we have, as required,

$$f(x^T) - f(x^*) \leqslant \frac{1}{T}\sum_{k=0}^{T-1}(f(x^{k+1}) - f^*) \leqslant \frac{L}{2T}\|x^0 - x^*\|^2. \qquad \square$$

4.4. Strongly Convex Case Recall that the definition (3.3.9) of strong convexity shows that f can be bounded below by a quadratic with Hessian γI. A strongly convex f with L-Lipschitz gradients is also bounded *above* by a similar quadratic (see (3.3.7)) differing only in the quadratic term, which becomes LI. From this "sandwich" effect, we derive a linear convergence rate for the gradient method, stated formally in the following theorem.

Theorem 4.4.1. *Suppose that f is Lipschitz continuously differentiable, satisfying (3.3.6), and strongly convex, satisfying (3.2.5) with modulus of convexity γ. Then f has a unique minimizer x^*, and the steepest descent method with stepsize $\alpha_k \equiv 1/L$ generates a sequence $\{x^k\}_{k=0}^{\infty}$ that satisfies*

$$f(x^{k+1}) - f(x^*) \leqslant \left(1 - \frac{\gamma}{L}\right)(f(x^k) - f(x^*)), \quad k = 0, 1, 2, \ldots.$$

Proof. Existence of the unique minimizer x^* follows from Theorem 3.3.14. Minimizing both sides of the inequality (3.3.9) with respect to y, we find that the minimizer on the left side is attained at $y = x^*$, while on the right side it is attained at $x - \nabla f(x)/\gamma$. Plugging these optimal values into (3.3.9), we obtain

$$\begin{aligned} &\min_y f(y) \geqslant \min_y f(x) + \nabla f(x)^T(y - x) + \frac{\gamma}{2}\|y - x\|^2 \\ \Rightarrow\quad &f(x^*) \geqslant f(x) - \nabla f(x)^T\left(\frac{1}{\gamma}\nabla f(x)\right) + \frac{\gamma}{2}\left\|\frac{1}{\gamma}\nabla f(x)\right\|^2 \\ \Rightarrow\quad &f(x^*) \geqslant f(x) - \frac{1}{2\gamma}\|\nabla f(x)\|^2. \end{aligned}$$

By rearrangement, we obtain

$$\|\nabla f(x)\|^2 \geqslant 2\gamma[f(x) - f(x^*)]. \tag{4.4.2}$$

By substituting (4.4.2) into our basic inequality (4.1.3), we obtain

$$f(x^{k+1}) = f\left(x^k - \frac{1}{L}\nabla f(x^k)\right) \leqslant f(x^k) - \frac{1}{2L}\|\nabla f(x^k)\|^2 \leqslant f(x^k) - \frac{\gamma}{L}(f(x^k) - f^*).$$

Subtracting f^* from both sides of this inequality yields the result. □

Note that After T steps, we have

$$f(x^T) - f^* \leqslant \left(1 - \frac{\gamma}{L}\right)^T (f(x^0) - f^*), \tag{4.4.3}$$

which is convergence of type (3.6.2) with constant $\sigma = \gamma/L$.

4.5. General Case: Line-Search Methods Returning to the case in which f has Lipschitz continuous gradients but is possibly nonconvex, we consider algorithms that take steps of the form (4.0.1), where d^k is a *descent direction*, that is, it makes a positive inner product with the negative gradient $-\nabla f(x^k)$, so that $\nabla f(x^k)^T d^k < 0$. This condition ensures that $f(x^k + \alpha d^k) < f(x^k)$ for sufficiently small positive values of step length α — we obtain improvement in f by taking small steps along d^k. (This claim follows from (3.3.3).) *Line-search methods* are built around this fundamental observation. By introducing additional conditions on d^k and α_k, that can be verified in practice with reasonable effort, we can establish a bound on decrease similar to (4.1.3) on each iteration, and thus a conclusion similar to that of Theorem 4.2.1.

We assume that d^k satisfies the following for some $\eta > 0$:

$$\nabla f(x^k)^T d^k \leqslant -\eta\|\nabla f(x^k)\|\|d^k\|. \tag{4.5.1}$$

For the steplength α_k, we assume the following *weak Wolfe* conditions hold, for some constants c_1 and c_2 with $0 < c_1 < c_2 < 1$:

$$f(x^k + \alpha_k d^k) \leqslant f(x^k) + c_1\alpha_k\nabla f(x^k)^T d^k \tag{4.5.2a}$$

$$\nabla f(x^k + \alpha_k d^k)^T d^k \geqslant c_2\nabla f(x^k)^T d^k. \tag{4.5.2b}$$

Condition (4.5.2a) is called "sufficient decrease;" it ensures descent at each step of at least a small fraction c_1 of the amount promised by the first-order Taylor-series expansion (3.3.3). Condition (4.5.2b) ensures that the directional derivative of f along the search direction d^k is significantly less negative at the chosen steplength α_k than at $\alpha = 0$. This condition ensures that the step is "not too short." It can be shown that it is always possible to find α_k that satisfies both conditions (4.5.2) simultaneously.

Line-search procedures, which are specialized optimization procedures for minimizing functions of one variable, have been devised to find such values efficiently; see [36, Chapter 3] for details.

For line-search methods of this type, we have the following generalization of Theorem 4.2.1.

Theorem 4.5.3. *Suppose that f is Lipschitz continuously differentiable, satisfying (3.3.6), and that f is bounded below by a constant $\bar{f}$. Consider the method that takes steps of the form (4.0.1), where d^k satisfies (4.5.1) for some $\eta > 0$ and the conditions (4.5.2) hold at all k, for some constants c_1 and c_2 with $0 < c_1 < c_2 < 1$. Then for any integer $T \geqslant 1$, we have*

$$\min_{0 \leqslant k \leqslant T-1} \|\nabla f(x^k)\| \leqslant \sqrt{\frac{L}{\eta^2 c_1(1-c_2)}} \sqrt{\frac{f(x^0)-\bar{f}}{T}}.$$

Proof. By combining the Lipschitz property (3.3.6) with (4.5.2b), we have

$$-(1-c_2)\nabla f(x^k)^T d^k \leqslant [\nabla f(x^k + \alpha_k d^k) - \nabla f(x^k)]^T d^k \leqslant L\alpha_k \|d^k\|^2.$$

By comparing the first and last terms in these inequalities, we obtain the following lower bound on α_k:

$$\alpha_k \geqslant -\frac{(1-c_2)}{L} \frac{\nabla f(x^k)^T d^k}{\|d^k\|^2}.$$

By substituting this bound into (4.5.2a), and using (4.5.1) and the step definition (4.0.1), we obtain

$$\begin{aligned} f(x^{k+1}) = f(x^k + \alpha_k d^k) &\leqslant f(x^k) + c_1 \alpha_k \nabla f(x^k)^T d^k \\ &\leqslant f(x^k) - \frac{c_1(1-c_2)}{L} \frac{(\nabla f(x^k)^T d^k)^2}{\|d^k\|^2} \\ &\leqslant f(x^k) - \frac{c_1(1-c_2)}{L} \eta^2 \|\nabla f(x^k)\|^2, \end{aligned} \tag{4.5.4}$$

which by rearrangement yields

$$\|\nabla f(x^k)\|^2 \leqslant \frac{L}{c_1(1-c_2)\eta^2} \left(f(x^k) - f(x^{k+1}) \right). \tag{4.5.5}$$

The result now follows as in the proof of Theorem 4.2.1. □

It follows by taking limits on both sides of (4.5.5) that

$$\lim_{k \to \infty} \|\nabla f(x^k)\| = 0, \tag{4.5.6}$$

and therefore all accumulation points $\bar{x}$ of the sequence $\{x^k\}$ generated by the algorithm (4.0.1) have $\nabla f(\bar{x}) = 0$. In the case of f convex, this condition guarantees that $\bar{x}$ is a solution of (1.0.1). When f is nonconvex, $\bar{x}$ may be a local minimum, but it may also be a saddle point or a local maximum.

The paper [29] uses the stable manifold theorem to show that line-search gradient methods are highly unlikely to converge to stationary points $\bar{x}$ at which some eigenvalues of the Hessian $\nabla^2 f(\bar{x})$ are negative. Although it is easy to construct examples for which such bad behavior occurs, it requires special choices of starting point x^0. Possibly the most obvious example is where $f(x_1, x_2) = x_1^2 - x_2^2$ starting from $x^0 = (1, 0)^T$, where $d^k = -\nabla f(x^k)$ at each k. For this example, all iterates have $x_2^k = 0$ and, under appropriate conditions, converge to the saddle point $\bar{x} = 0$. Any starting point with $x_2^0 \neq 0$ cannot converge to 0, in fact, it is easy to see that x_2^k diverges away from 0.

4.6. Conditional Gradient Method The conditional gradient approach, often known as "Frank-Wolfe" after the authors who devised it [24], is a method for convex nonlinear optimization over compact convex sets. This is the problem

$$\min_{x\in\Omega} f(x), \tag{4.6.1}$$

(see earlier discussion around (3.2.2)), where Ω is a compact convex set and f is a convex function whose gradient is Lipschitz continuously differentiable in a neighborhood of Ω, with Lipschitz constant L. We assume that Ω has diameter D, that is, $\|x-y\| \leqslant D$ for all $x, y \in \Omega$.

The conditional gradient method replaces the objective in (4.6.1) at each iteration by a linear Taylor-series approximation around the current iterate x^k, and minimizes this linear objective over the original constraint set Ω. It then takes a step from x^k towards the minimizer of this linearized subproblem. The full method is as follows:

$$v^k := \arg\min_{v\in\Omega} v^T\nabla f(x^k); \tag{4.6.2a}$$

$$x^{k+1} := x^k + \alpha_k(v^k - x^k), \quad \alpha_k := \frac{2}{k+2}. \tag{4.6.2b}$$

The method has a sublinear convergence rate, as we show below, and indeed requires many iterations in practice to obtain an accurate solution. Despite this feature, it makes sense in many interesting applications, because the subproblems (4.6.2a) can be solved very cheaply in some settings, and because highly accurate solutions are not required in some applications.

We have the following result for sublinear convergence of the conditional gradient method.

Theorem 4.6.3. *Under the conditions above, where L is the Lipschitz constant for ∇f on an open neighborhood of Ω and D is the diameter of Ω, the conditional gradient method (4.6.2) applied to (4.6.1) satisfies*

$$f(x^k) - f(x^*) \leqslant \frac{2LD^2}{k+2}, \quad k = 1, 2, \ldots, \tag{4.6.4}$$

where x^* is any solution of (4.6.1).

Proof. Setting $x = x^k$ and $y = x^{k+1} = x^k + \alpha_k(v^k - x^k)$ in (3.3.7), we have

$$\begin{aligned} f(x^{k+1}) &\leqslant f(x^k) + \alpha_k\nabla f(x^k)^T(v^k - x^k) + \frac{1}{2}\alpha_k^2 L\|v^k - x^k\|^2 \\ &\leqslant f(x^k) + \alpha_k\nabla f(x^k)^T(v^k - x^k) + \frac{1}{2}\alpha_k^2 LD^2, \end{aligned} \tag{4.6.5}$$

where the second inequality comes from the definition of D. For the first-order term, we have since v^k solves (4.6.2a) and x^* is feasible for (4.6.2a) that

$$\nabla f(x^k)^T(v^k - x^k) \leqslant \nabla f(x^k)^T(x^* - x^k) \leqslant f(x^*) - f(x^k).$$

By substituting in (4.6.5) and subtracting $f(x^*)$ from both sides, we obtain

$$f(x^{k+1}) - f(x^*) \leqslant (1-\alpha_k)[f(x^k) - f(x^*)] + \frac{1}{2}\alpha_k^2 LD^2. \tag{4.6.6}$$

We now apply an inductive argument. For $k = 0$, we have $\alpha_0 = 1$ and

$$f(x^1) - f(x^*) \leqslant \frac{1}{2}LD^2 < \frac{2}{3}LD^2,$$

so that (4.6.4) holds in this case. Supposing that (4.6.4) holds for some value of k, we aim to show that it holds for $k+1$ too. We have

$$\begin{aligned}
& f(x^{k+1}) - f(x^*) \\
& \leqslant \left(1 - \frac{2}{k+2}\right)[f(x^k) - f(x^*)] + \frac{1}{2}\frac{4}{(k+2)^2}LD^2 && \text{from (4.6.6), (4.6.2b)} \\
& \leqslant LD^2\left[\frac{2k}{(k+2)^2} + \frac{2}{(k+2)^2}\right] && \text{from (4.6.4)} \\
& = 2LD^2\frac{(k+1)}{(k+2)^2} \\
& = 2LD^2\frac{k+1}{k+2}\frac{1}{k+2} \\
& \leqslant 2LD^2\frac{k+2}{k+3}\frac{1}{k+2} = \frac{2LD^2}{k+3},
\end{aligned}$$

as required. □

5. Prox-Gradient Methods

We now describe an elementary but powerful approach for solving the regularized optimization problem

$$\min_{x \in \mathbb{R}^n} \phi(x) := f(x) + \lambda\psi(x), \tag{5.0.1}$$

where f is a smooth convex function, ψ is a convex regularization function (known simply as the "regularizer"), and $\lambda \geqslant 0$ is a regularization parameter. The technique we describe here is a natural extension of the steepest-descent approach, in that it reduces to the steepest-descent method analyzed in Theorems 4.3.1 and 4.4.1 applied to f when the regularization term is not present ($\lambda = 0$). It is useful when the regularizer ψ has a simple structure that is easy to account for explicitly, as is true for many regularizers that arise in data analysis, such as the ℓ_1 function ($\psi(x) = \|x\|_1$) of the indicator function for a simple set Ω ($\psi(x) = I_\Omega(x)$), such as a box $\Omega = [l_1, u_1] \otimes [l_2, u_2] \otimes \ldots \otimes [l_n, u_n]$. For such regularizers, the proximal operators can be computed explicitly and efficiently.[2]

Each step of the algorithm is defined as follows:

$$x^{k+1} := \text{prox}_{\alpha_k\lambda\psi}(x^k - \alpha_k\nabla f(x^k)), \tag{5.0.2}$$

for some steplength $\alpha_k > 0$, and the prox operator defined in (3.5.3). By substituting into this definition, we can verify that x^{k+1} is the solution of an approximation to the objective ϕ of (5.0.1), namely:

$$x^{k+1} := \arg\min_z \nabla f(x^k)^T(z - x^k) + \frac{1}{2\alpha_k}\|z - x^k\|^2 + \lambda\psi(z). \tag{5.0.3}$$

[2]For the analysis of this section I am indebted to class notes of L. Vandenberghe, from 2013-14.

One way to verify this equivalence is to note that the objective in (5.0.3) can be written as

$$\frac{1}{\alpha_k}\left\{\frac{1}{2}\left\|z-(x^k-\alpha_k\nabla f(x^k))\right\|^2+\alpha_k\lambda\psi(x)\right\},$$

(modulo a term $\alpha_k\|\nabla f(x^k)\|^2$ that does not involve z). The subproblem objective in (5.0.3) consists of a linear term $\nabla f(x^k)^T(z-x^k)$ (the first-order term in a Taylor-series expansion), a proximality term $\frac{1}{2\alpha_k}\|z-x^k\|^2$ that becomes more strict as $\alpha_k \downarrow 0$, and the regularization term $\lambda\psi(x)$ in unaltered form. When $\lambda = 0$, we have $x^{k+1} = x^k - \alpha_k\nabla f(x^k)$, so the iteration (5.0.2) (or (5.0.3)) reduces to the usual steepest-descent approach discussed in Section 4 in this case. It is useful to continue thinking of α_k as playing the role of a line-search parameter, though here the line search is expressed implicitly through a proximal term.

We will demonstrate convergence of the method (5.0.2) at a sublinear rate, for functions f whose gradients satisfy a Lipschitz continuity property with Lipschitz constant L (see (3.3.6)), and for the constant steplength choice $\alpha_k = 1/L$. The proof makes use of a "gradient map" defined by

$$G_\alpha(x) := \frac{1}{\alpha}\left(x-\text{prox}_{\alpha\lambda\psi}(x-\alpha\nabla f(x))\right). \tag{5.0.4}$$

By comparing with (5.0.2), we see that this map defines the step taken at iteration k:

$$x^{k+1} = x^k - \alpha_k G_{\alpha_k}(x^k) \quad\Leftrightarrow\quad G_{\alpha_k}(x^k) = \frac{1}{\alpha_k}(x^k - x^{k+1}). \tag{5.0.5}$$

The following technical lemma reveals some useful properties of $G_\alpha(x)$.

Lemma 5.0.6. *Suppose that in problem* (5.0.1), ψ *is a closed convex function and that* f *is convex with Lipschitz continuous gradient on* $\mathbb{R}^n$*, with Lipschitz constant* L*. Then for the definition* (5.0.4) *with* $\alpha > 0$*, the following claims are true.*

(a) $G_\alpha(x) \in \nabla f(x) + \lambda\partial\psi(x-\alpha G_\alpha(x))$.

(b) *For any* z*, and any* $\alpha \in (0, 1/L]$*, we have that*

$$\phi(x-\alpha G_\alpha(x)) \leqslant \phi(z) + G_\alpha(x)^T(x-z) - \frac{\alpha}{2}\|G_\alpha(x)\|^2.$$

Proof. For part (a), we use the optimality property (3.5.4) of the prox operator, and make the following substitutions: $x-\alpha\nabla f(x)$ for "x", $\alpha\lambda$ for "λ", and ψ for "h" to obtain

$$0 \in \alpha\lambda\partial\psi(\text{prox}_{\alpha\lambda\psi}(x-\alpha\nabla f(x))) + (\text{prox}_{\alpha\lambda\psi}(x-\alpha\nabla f(x)) - (x-\alpha\nabla f(x)).$$

We make the substitution $\text{prox}_{\alpha\lambda\psi}(x-\alpha\nabla f(x)) = x-\alpha G_\alpha(x)$, using definition (5.0.4), to obtain

$$0 \in \alpha\lambda\partial\psi(x-\alpha G_\alpha(x)) - \alpha(G_\alpha(x) - \nabla f(x)),$$

and the result follows when we divide by α.

For (b), we start with the following consequence of Lipschitz continuity of ∇f, from Lemma 3.3.10:

$$f(y) \leqslant f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2.$$

By setting $y = x - \alpha G_\alpha(x)$, for any $\alpha \in (0, 1/L]$, we have

$$\begin{aligned} f(x - \alpha G_\alpha(x)) &\leqslant f(x) - \alpha G_\alpha(x)^T \nabla f(x) + \frac{L\alpha^2}{2} \|G_\alpha(x)\|^2 \\ &\leqslant f(x) - \alpha G_\alpha(x)^T \nabla f(x) + \frac{\alpha}{2} \|G_\alpha(x)\|^2. \end{aligned} \tag{5.0.7}$$

(The second inequality uses $\alpha \in (0, 1/L]$.) We also have by convexity of f and ψ that for any z and any $v \in \partial\psi(x - \alpha G_\alpha(x)$ the following are true:

$$\begin{aligned} f(z) &\geqslant f(x) + \nabla f(x)^T (z - x), \\ \psi(z) &\geqslant \psi(x - \alpha G_\alpha(x)) + v^T (z - (x - \alpha G_\alpha(x))). \end{aligned} \tag{5.0.8}$$

From part (a) that $v = (G_\alpha(x) - \nabla f(x))/\lambda \in \partial\psi(x - \alpha G_\alpha(x))$. Making this choice of v in (5.0.8) and using (5.0.7) we have for any $\alpha \in (0, 1/L]$ that

$$\begin{aligned} &\phi(x - \alpha G_\alpha(x)) \\ &= f(x - \alpha G_\alpha(x)) + \lambda\psi(x - \alpha G_\alpha(x)) \\ &\leqslant f(x) - \alpha G_\alpha(x)^T \nabla f(x) + \frac{\alpha}{2} \|G_\alpha(x)\|^2 + \lambda\psi(x - \alpha G_\alpha(x)) \quad \text{(from (5.0.7))} \\ &\leqslant f(z) + \nabla f(x)^T (x - z) - \alpha G_\alpha(x)^T \nabla f(x) + \frac{\alpha}{2} \|G_\alpha(x)\|^2 \\ &\quad + \lambda\psi(z) + (G_\alpha(x) - \nabla f(x))^T (x - \alpha G_\alpha(x) - z) \qquad \text{(from (5.0.8))} \\ &= f(z) + \lambda\psi(z) + G_\alpha(x)^T (x - z) - \frac{\alpha}{2} \|G_\alpha(x)\|^2, \end{aligned}$$

where the last equality follows from cancellation of several terms in the previous line. Thus (b) is proved. □

Theorem 5.0.9. *Suppose that in problem (5.0.1), ψ is a closed convex function and that f is convex with Lipschitz continuous gradient on $\mathbb{R}^n$, with Lipschitz constant L. Suppose that (5.0.1) attains a minimizer x^* (not necessarily unique) with optimal objective value ϕ^*. Then if $\alpha_k = 1/L$ for all k in (5.0.2), we have*

$$\phi(x^k) - \phi^* \leqslant \frac{L\|x^0 - x^*\|^2}{2k}, \quad k = 1, 2, \ldots .$$

Proof. Since $\alpha_k = 1/L$ satisfies the conditions of Lemma 5.0.6, we can use part (b) of this result to show that the sequence $\{\phi(x^k)\}$ is decreasing and that the distance to the optimum x^* also decreases at each iteration. Setting $x = z = x^k$ and $\alpha = \alpha_k$ in Lemma 5.0.6, and recalling (5.0.5), we have

$$\phi(x^{k+1}) = \phi(x^k - \alpha_k G_{\alpha_k}(x^k)) \leqslant \phi(x^k) - \frac{\alpha_k}{2} \|G_{\alpha_k}(x^k)\|^2,$$

justifying the first claim. For the second claim, we have by setting $x = x^k$, $\alpha = \alpha_k$, and $z = x^*$ in Lemma 5.0.6 that

$$\begin{aligned} 0 \leqslant \phi(x^{k+1}) - \phi^* &= \phi(x^k - \alpha_k G_{\alpha_k}(x^k)) - \phi^* \\ &\leqslant G_{\alpha_k}(x^k)^T (x^k - x^*) - \frac{\alpha_k}{2} \|G_{\alpha_k}(x^k)\|^2 \\ &= \frac{1}{2\alpha_k} \left(\|x^k - x^*\|^2 - \|x^k - x^* - \alpha_k G_{\alpha_k}(x^k)\|^2 \right) \\ &= \frac{1}{2\alpha_k} \left(\|x^k - x^*\|^2 - \|x^{k+1} - x^*\|^2 \right), \end{aligned} \tag{5.0.10}$$

from which $\|x^{k+1} - x^*\| \leqslant \|x^k - x^*\|$ follows.

By setting $\alpha_k = 1/L$ in (5.0.10), and summing over $k = 0, 1, 2, \ldots, K-1$, we obtain from a telescoping sum on the right-hand side that

$$\sum_{k=0}^{K-1} (\phi(x^{k+1}) - \phi^*) \leqslant \frac{L}{2} \left(\|x^0 - x^*\|^2 - \|x^K - x^*\|^2 \right) \leqslant \frac{L}{2} \|x^0 - x^*\|^2.$$

By monotonicity of $\{\phi(x^k)\}$, we have

$$K(\phi(x^K) - \phi^*) \leqslant \sum_{k=0}^{K-1} (\phi(x^{k+1}) - \phi^*).$$

The result follows immediately by combining these last two expressions. □

6. Accelerating Gradient Methods

We showed in Section 4 that the basic steepest descent method for solving (1.0.1) for smooth f converges sublinearly at a $1/k$ rate when f is convex, and linearly at a rate of $(1 - \gamma/L)$ when f is strongly convex, satisfying (3.3.13) for positive γ and L. We show in this section that by using the gradient information in a more clever way, faster convergence rates can be attained.

The key idea is *momentum*. In iteration k of a momentum method, we tend to continue moving along the *previous* search direction at each iteration, making a small adjustment toward the negative gradient $-\nabla f$ evaluated at x^k or a nearby point. (Steepest descent simply uses $-\nabla f(x^k)$ as the search direction.) Although not obvious at first, there is some intuition behind the momentum idea. The step taken at the previous iterate x^{k-1} was based on negative gradient information at that iteration, along with the search direction from the iteration prior to that one, namely, x^{k-2}. By continuing this line of reasoning backwards, we see that the previous step is a linear combination of all the gradient information that we have encountered at all iterates so far, going back to the initial iterate x^0. If this information is aggregated properly, it can produce a richer overall picture of the function than the latest negative gradient alone, and thus has the potential to yield better convergence.

Sure enough, several intricate methods that use the momentum idea have been proposed, and have been widely successful. These methods are often called *accelerated gradient methods*. A major contributor in this area is Yuri Nesterov, dating to his seminal contribution in 1983 [33] and explicated further in his book [34] and other publications. Another key contribution is [3], which derived an accelerated method for the regularized case (1.0.2).

6.1. Heavy-Ball Method Possibly the most elementary method of momentum type is the *heavy-ball* method of Polyak [37]; see also [38]. Each iteration of this method has the form

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k) + \beta_k (x^k - x^{k-1}), \tag{6.1.1}$$

where α_k and β_k are positive scalars. That is, a momentum term $\beta_k(x^k - x^{k-1})$ is added to the usual steepest descent update. Although this method can be applied to any smooth convex f (and even to nonconvex functions), the convergence analysis is most straightforward for the special case of strongly convex *quadratic* functions (see [38]). (This analysis also suggests appropriate values for the step lengths α_k and β_k.) Consider the function

$$\min_{x \in \mathbb{R}^n} f(x) := \frac{1}{2} x^T A x - b^T x, \tag{6.1.2}$$

where the (constant) Hessian A has eigenvalues in the range $[\gamma, L]$, with $0 < \gamma \leqslant L$. For the following constant choices of steplength parameters:

$$\alpha_k = \alpha := \frac{4}{(\sqrt{L} + \sqrt{\gamma})^2}, \quad \beta_k = \beta := \frac{\sqrt{L} - \sqrt{\gamma}}{\sqrt{L} + \sqrt{\gamma}},$$

it can be shown that $\|x^k - x^*\| \leqslant C\beta^k$, for some (possibly large) constant C. We can use (3.3.7) to translate this into a bound on the function error, as follows:

$$f(x^k) - f(x^*) \leqslant \frac{L}{2}\|x^k - x^*\|^2 \leqslant \frac{LC^2}{2}\beta^{2k},$$

allowing a direct comparison with the rate (4.4.3) for the steepest descent method. If we suppose that $L \gg \gamma$, we have

$$\beta \approx 1 - 2\sqrt{\frac{\gamma}{L}},$$

so that we achieve approximate convergence $f(x^k) - f(x^*) \leqslant \epsilon$ (for small positive ϵ) in $O(\sqrt{L/\gamma}\log(1/\epsilon))$ iterations, compared with $O((L/\gamma)\log(1/\epsilon))$ for steepest descent — a significant improvement.

The heavy-ball method is fundamental, but several points should be noted. First, the analysis for convex quadratic f is based on linear algebra arguments, and does not generalize to general strongly convex nonlinear functions. Second, the method requires knowledge of γ and L, for the purposes of defining parameters α and β. Third, it is not a descent method; we usually have $f(x^{k+1}) > f(x^k)$ for many k. These properties are not specific to the heavy-ball method — some of them are shared by other methods that use momentum.

6.2. Conjugate Gradient The conjugate gradient method for solving linear systems $Ax = b$ (or, equivalently, minimizing the convex quadratic (6.1.2)) where A is symmetric positive definite, is one of the most important algorithms in computational science. Though invented earlier than the other algorithms discussed in this section (see [27]) and motivated in a different way, conjugate gradient clearly makes use of momentum. Its steps have the form

$$x^{k+1} = x^k + \alpha_k p^k, \quad \text{where } p^k = -\nabla f(x^k) + \xi_k p^{k-1}, \tag{6.2.1}$$

for some choices of α_k and ξ_k, which is identical to (6.1.1) when we define β_k appropriately. For convex, strongly quadratic problems (6.1.2), conjugate gradient has excellent properties. It does not require prior knowledge of the range $[\gamma, L]$ of

the eigenvalue spectrum of A, choosing the steplengths α_k and ξ_k in an adaptive fashion. (In fact, α_k is chosen to be the exact minimizer along the search direction p_k.) The main arithmetic operation per iteration is one matrix-vector multiplication involving A, the same cost as a gradient evaluation for f in (6.1.2). Most importantly, there is a rich convergence theory, that characterizes convergence in terms of the properties of the full spectrum of A (not just its extreme elements), showing in particular that good approximate solutions can be obtained quickly if the eigenvalues are clustered. Convergence to an exact solution of (6.1.2) in at most n iterations is guaranteed (provided, naturally, that the arithmetic is carried out exactly).

There has been much work over the years on extending the conjugate gradient method to general smooth functions f. Few of the theoretical properties for the quadratic case carry over to the nonlinear setting, though several results are known; see [36, Chapter 5], for example. Such "nonlinear" conjugate gradient methods vary in the accuracy with which they perform the line search for α_k in (6.2.1) and — more fundamentally — in the choice of ξ_k. The latter is done in a way that ensures that each search direction p^k is a descent direction. In some methods, ξ_k is set to zero on some iterations, which causes the method to take a steepest descent step, effectively "restarting" the conjugate gradient method at the latest iterate.

Despite these qualifications, nonlinear conjugate gradient is quite commonly used in practice, because of its minimal storage requirements and the fact that it requires only one gradient evaluation per iteration. Its popularity has been eclipsed in recent years by the limited-memory quasi-Newton method L-BFGS [30], [36, Section 7.2], which requires more storage (though still $O(n)$) and is similarly economical and easy to implement.

6.3. Nesterov's Accelerated Gradient: Weakly Convex Case We now describe Nesterov's method for (1.0.1) and prove its convergence — sublinear at a $1/k^2$ rate — for the case of f convex with Lipschitz continuous gradients satisfying (3.3.6). Each iteration of this method has the form

$$x^{k+1} = x^k - \alpha_k \nabla f\left(x^k + \beta_k (x^k - x^{k-1})\right) + \beta_k (x^k - x^{k-1}), \tag{6.3.1}$$

for choices of the parameters α_k and β_k to be defined. Note immediately the similarity to the heavy-ball formula (6.1.1). The only difference is that the extrapolation step $x^k \to x^k + \beta_k(x^k - x^{k-1})$ is taken before evaluation of the gradient ∇f in (6.3.1), whereas in (6.1.1) the gradient is simply evaluated at x^k. It is convenient for purposes of analysis (and implementation) to introduce an auxiliary sequence $\{y^k\}$, fix $\alpha_k \equiv 1/L$, and rewrite the update (6.3.1) as follows:

$$x^{k+1} = y^k - \frac{1}{L}\nabla f(y^k), \tag{6.3.2a}$$

$$y^{k+1} = x^{k+1} + \beta_{k+1}(x^{k+1} - x^k), \quad k = 0, 1, 2, \ldots, \tag{6.3.2b}$$

where we initialize at an arbitrary y^0 and set $x^0 = y^0$. We define β_k with reference to another scalar sequence λ_k in the following manner:

$$\lambda_0 = 0, \quad \lambda_{k+1} = \frac{1}{2}\left(1 + \sqrt{1 + 4\lambda_k^2}\right), \quad \beta_k = \frac{\lambda_k - 1}{\lambda_{k+1}}. \tag{6.3.3}$$

Since $\lambda_k \geqslant 1$ for $k = 1, 2, \ldots$, we have $\beta_{k+1} \geqslant 0$ for $k = 0, 1, 2, \ldots$. It also follows from the definition of λ_{k+1} that

$$\lambda_{k+1}^2 - \lambda_{k+1} = \lambda_k^2. \tag{6.3.4}$$

We have the following result for convergence of Nesterov's scheme on general convex functions. We prove it using an argument from [3], as reformulated in [7, Section 3.7]. The analysis is famously technical, and intuition is hard to come by. Some recent progress has been made in deriving algorithms similar to (6.3.2) that have a plausible geometric or algebraic motivation; see [8, 21].

Theorem 6.3.5. *Suppose that f in (1.0.1) is convex, with ∇f Lipschitz continuously differentiable with constant L (as in (3.3.6)) and that the minimum of f is attained at x^*, with $f^* := f(x^*)$. Then the method defined by (6.3.2), (6.3.3) with $x^0 = y^0$ yields an iteration sequence $\{x^k\}$ with the following property:*

$$f(x^T) - f^* \leqslant \frac{2L\|x^0 - x^*\|^2}{(T+1)^2}, \quad T = 1, 2, \ldots .$$

Proof. From convexity of f and (3.3.7), we have for any x and y that

$$\begin{aligned} & f(y - \nabla f(y)/L) - f(x) \\ & \leqslant f(y - \nabla f(y)/L) - f(y) + \nabla f(y)^T (y - x) \\ & \leqslant \nabla f(y)^T (y - \nabla f(y)/L - y) + \frac{L}{2}\|y - \nabla f(y)/L - y\|^2 + \nabla f(y)^T (y - x) \\ & = -\frac{1}{2L}\|\nabla f(y)\|^2 + \nabla f(y)^T (y - x). \end{aligned} \tag{6.3.6}$$

Setting $y = y^k$ and $x = x^k$ in this bound, we obtain

$$\begin{aligned} f(x^{k+1}) - f(x^k) &= f(y^k - \nabla f(y^k)/L) - f(x^k) \\ &\leqslant -\frac{1}{2L}\|\nabla f(y^k)\|^2 + \nabla f(y^k)^T (y^k - x^k) \\ &= -\frac{L}{2}\|x^{k+1} - y^k\|^2 - L(x^{k+1} - y^k)^T (y^k - x^k). \end{aligned} \tag{6.3.7}$$

We now set $y = y^k$ and $x = x^*$ in (6.3.6), and use (6.3.2a) to obtain

$$f(x^{k+1}) - f(x^*) \leqslant -\frac{L}{2}\|x^{k+1} - y^k\|^2 - L(x^{k+1} - y^k)^T (y^k - x^*). \tag{6.3.8}$$

Introducing notation $\delta_k := f(x^k) - f(x^*)$, we multiply (6.3.7) by $\lambda_{k+1} - 1$ and add it to (6.3.8) to obtain

$$\begin{aligned} & (\lambda_{k+1} - 1)(\delta_{k+1} - \delta_k) + \delta_{k+1} \\ & \leqslant -\frac{L}{2}\lambda_{k+1}\|x^{k+1} - y^k\|^2 - L(x^{k+1} - y^k)^T (\lambda_{k+1} y^k - (\lambda_{k+1} - 1)x^k - x^*). \end{aligned}$$

We multiply this bound by λ_{k+1}, and use (6.3.4) to obtain

$$\begin{aligned}(6.3.9)\quad & \lambda_{k+1}^2\delta_{k+1} - \lambda_k^2\delta_k \\ &\leqslant -\frac{L}{2}\left[\|\lambda_{k+1}(x^{k+1}-y^k)\|^2 + 2\lambda_{k+1}(x^{k+1}-y^k)^T(\lambda_{k+1}y^k - (\lambda_{k+1}-1)x^k - x^*)\right] \\ &= -\frac{L}{2}\left[\|\lambda_{k+1}x^{k+1} - (\lambda_{k+1}-1)x^k - x^*\|^2 - \|\lambda_{k+1}y^k - (\lambda_{k+1}-1)x^k - x^*\|^2\right],\end{aligned}$$

where in the final equality we used the identity $\|a\|^2 + 2a^Tb = \|a+b\|^2 - \|b\|^2$. By multiplying (6.3.2b) by λ_{k+2}, and using $\lambda_{k+2}\beta_{k+1} = \lambda_{k+1} - 1$ from (6.3.3), we have

$$\begin{aligned}\lambda_{k+2}y^{k+1} &= \lambda_{k+2}x^{k+1} + \lambda_{k+2}\beta_{k+1}(x^{k+1}-x^k) \\ &= \lambda_{k+2}x^{k+1} + (\lambda_{k+1}-1)(x^{k+1}-x^k).\end{aligned}$$

By rearranging this equality, we have

$$\lambda_{k+1}x^{k+1} - (\lambda_{k+1}-1)x^k = \lambda_{k+2}y^{k+1} - (\lambda_{k+2}-1)x^{k+1}.$$

By substituting into the first term on the right-hand side of (6.3.9), and using the definition

$$(6.3.10)\qquad u^k := \lambda_{k+1}y^k - (\lambda_{k+1}-1)x^k - x^*,$$

we obtain

$$\lambda_{k+1}^2\delta_{k+1} - \lambda_k^2\delta_k \leqslant -\frac{L}{2}(\|u^{k+1}\|^2 - \|u^k\|^2).$$

By summing both sides of this inequality over $k = 0, 1, \ldots, T-1$, and using $\lambda_0 = 0$, we obtain

$$\lambda_T^2\delta_T \leqslant \frac{L}{2}(\|u^0\|^2 - \|u^T\|^2) \leqslant \frac{L}{2}\|x^0 - x^*\|^2,$$

so that

$$(6.3.11)\qquad \delta_T = f(x^T) - f(x^*) \leqslant \frac{L\|x^0 - x^*\|^2}{2\lambda_T^2}.$$

A simple induction confirms that $\lambda_k \geqslant (k+1)/2$ for $k = 1, 2, \ldots$, and the claim of the theorem follows by substituting this bound into (6.3.11). □

6.4. Nesterov's Accelerated Gradient: Strongly Convex Case We turn now to Nesterov's approach for smooth strongly convex functions, which satisfy (3.2.5) with $\gamma > 0$. Again, we follow the proof in [7, Section 3.7], which is based on the analysis in [34]. The method uses the same update formula (6.3.2) as in the weakly convex case, and the same initialization, but with a different choice of β_{k+1}, namely:

$$(6.4.1)\qquad \beta_{k+1} \equiv \frac{\sqrt{L} - \sqrt{\gamma}}{\sqrt{L} + \sqrt{\gamma}} = \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}.$$

The condition measure κ is defined in (3.3.12). We prove the following convergence result.

Theorem 6.4.2. *Suppose that f is such that ∇f is Lipschitz continuously differentiable with constant L, and that it is strongly convex with modulus of convexity γ and unique*

minimizer x^*. Then the method (6.3.2), (6.4.1) with starting point $x^0 = y^0$ satisfies

$$f(x^T) - f(x^*) \leqslant \frac{L+\gamma}{2}\|x^0 - x^*\|^2 \left(1 - \frac{1}{\sqrt{\kappa}}\right)^T, \quad T = 1, 2, \ldots .$$

Proof. The proof makes use of a family of strongly convex functions $\Phi_k(z)$ defined inductively as follows:

$$\Phi_0(z) = f(y^0) + \frac{\gamma}{2}\|z - y^0\|^2, \tag{6.4.3a}$$

$$\Phi_{k+1}(z) = (1 - 1/\sqrt{\kappa})\Phi_k(z) + \frac{1}{\sqrt{\kappa}}\left(f(y^k) + \nabla f(y^k)^T(z - y^k) + \frac{\gamma}{2}\|z - y^k\|^2\right). \tag{6.4.3b}$$

Each $\Phi_k(\cdot)$ is a quadratic, and an inductive argument shows that $\nabla^2\Phi_k(z) = \gamma I$ for all k and all z. Thus, each Φ_k has the form

$$\Phi_k(z) = \Phi_k^* + \frac{\gamma}{2}\|z - v^k\|^2, \quad k = 0, 1, 2, \ldots, \tag{6.4.4}$$

where v^k is the minimizer of $\Phi_k(\cdot)$ and Φ_k^* is its optimal value. (From (6.4.3a), we have $v^0 = y^0$.) We note too that Φ_k becomes a tighter *overapproximation* to f as $k \to \infty$. To show this, we use (3.3.9) to replace the final term in parentheses in (6.4.3b) by $f(z)$, then subtract $f(z)$ from both sides of (6.4.3b) to obtain

$$\Phi_{k+1}(z) - f(z) \leqslant (1 - 1/\sqrt{\kappa})(\Phi_k(z) - f(z)). \tag{6.4.5}$$

In the remainder of the proof, we show that the following bound holds:

$$f(x^k) \leqslant \min_z \Phi_k(z) = \Phi_k^*, \quad k = 0, 1, 2, \ldots . \tag{6.4.6}$$

The upper bound in Lemma 3.3.10 for $x = x^*$ gives $f(z) - f(x^*) \leqslant (L/2)\|z - x^*\|^2$. By combining this bound with (6.4.5) and (6.4.6), we have

$$\begin{aligned} f(x^k) - f(x^*) &\leqslant \Phi_k^* - f(x^*) \\ &\leqslant \Phi_k(x^*) - f(x^*) \\ &\leqslant (1 - 1/\sqrt{\kappa})^k(\Phi_0(x^*) - f(x^*)) \\ &\leqslant (1 - 1/\sqrt{\kappa})^k[(\Phi_0(x^*) - f(x^0)) + (f(x^0) - f(x^*))] \\ &\leqslant (1 - 1/\sqrt{\kappa})^k \frac{\gamma + L}{2}\|x^0 - x^*\|^2. \end{aligned} \tag{6.4.7}$$

The proof is completed by establishing (6.4.6), by induction on k. Since $x^0 = y^0$, it holds by definition at $k = 0$. By using step formula (6.3.2a), the convexity property (3.3.8) (with $x = y^k$), and the inductive hypothesis, we have

$$\begin{aligned} & f(x^{k+1}) \\ &\leqslant f(y^k) - \frac{1}{2L}\|\nabla f(y^k)\|^2 \\ &= (1 - 1/\sqrt{\kappa})f(x^k) + (1 - 1/\sqrt{\kappa})(f(y^k) - f(x^k)) + f(y^k)/\sqrt{\kappa} - \frac{1}{2L}\|\nabla f(y^k)\|^2 \\ &\leqslant (1 - 1/\sqrt{\kappa})\Phi_k^* + (1 - 1/\sqrt{\kappa})\nabla f(y^k)^T(y^k - x^k) + f(y^k)/\sqrt{\kappa} - \frac{1}{2L}\|\nabla f(y^k)\|^2. \end{aligned} \tag{6.4.8}$$

Thus the claim is established (and the theorem is proved) if we can show that the right-hand side in (6.4.8) is bounded above by Φ^*_{k+1}.

Recalling the observation (6.4.4), we have by taking derivatives of both sides of (6.4.3b) with respect to z that

$$\nabla\Phi_{k+1}(z) = \gamma(1-1/\sqrt{\kappa})(z-v^k) + \nabla f(y^k)/\sqrt{\kappa} + \gamma(z-y^k)/\sqrt{\kappa}. \tag{6.4.9}$$

Since v^{k+1} is the minimizer of Φ_{k+1} we can set $\nabla\Phi_{k+1}(v^{k+1}) = 0$ in (6.4.9) to obtain

$$v^{k+1} = (1-1/\sqrt{\kappa})v^k + y^k/\sqrt{\kappa} - \nabla f(y^k)/(\gamma\sqrt{\kappa}). \tag{6.4.10}$$

By subtracting y^k from both sides of this expression, and taking $\|\cdot\|^2$ of both sides, we obtain

$$\begin{aligned}\|v^{k+1}-y^k\|^2 = (1-1/\sqrt{\kappa})^2\|y^k-v^k\|^2 + \|\nabla f(y^k)\|^2/(\gamma^2\kappa)\\ -2(1-1/\sqrt{\kappa})/(\gamma\sqrt{\kappa})\nabla f(y^k)^T(v^k-y^k).\end{aligned} \tag{6.4.11}$$

By evaluating Φ_{k+1} at $z = y^k$, using both (6.4.4) and (6.4.3b), we obtain

$$\begin{aligned}&\Phi^*_{k+1} + \frac{\gamma}{2}\|y^k - v^{k+1}\|^2\\ &= (1-1/\sqrt{\kappa})\Phi_k(y^k) + f(y^k)/\sqrt{\kappa}\\ &= (1-1/\sqrt{\kappa})\Phi^*_k + \frac{\gamma}{2}(1-1/\sqrt{\kappa})\|y^k-v^k\|^2 + f(y^k)/\sqrt{\kappa}.\end{aligned} \tag{6.4.12}$$

By substituting (6.4.11) into (6.4.12), we obtain

$$\begin{aligned}\Phi^*_{k+1} &= (1-1/\sqrt{\kappa})\Phi^*_k + f(y^k)/\sqrt{\kappa} + \gamma(1-1/\sqrt{\kappa})/(2\sqrt{\kappa})\|y^k-v^k\|^2\\ &\quad - \frac{1}{2L}\|\nabla f(y^k)\|^2 + (1-1/\sqrt{\kappa})\nabla f(y^k)^T(v^k-y^k)/\sqrt{\kappa}\\ &\geqslant (1-1/\sqrt{\kappa})\Phi^*_k + f(y^k)/\sqrt{\kappa}\\ &\quad - \frac{1}{2L}\|\nabla f(y^k)\|^2 + (1-1/\sqrt{\kappa})\nabla f(y^k)^T(v^k-y^k)/\sqrt{\kappa},\end{aligned} \tag{6.4.13}$$

where we simply dropped a nonnegative term from the right-hand side to obtain the inequality. The final step is to show that

$$v^k - y^k = \sqrt{\kappa}(y^k - x^k), \tag{6.4.14}$$

which we do by induction. Note that $v^0 = x^0 = y^0$, so the claim holds for $k = 0$. We have

$$\begin{aligned}v^{k+1} - y^{k+1} &= (1-1/\sqrt{\kappa})v^k + y^k/\sqrt{\kappa} - \nabla f(y^k)/(\gamma\sqrt{\kappa}) - y^{k+1}\\ &= \sqrt{\kappa}y^k - (\sqrt{\kappa}-1)x^k - \sqrt{\kappa}\nabla f(y^k)/L - y^{k+1}\\ &= \sqrt{\kappa}x^{k+1} - (\sqrt{\kappa}-1)x^k - y^{k+1}\\ &= \sqrt{\kappa}(y^{k+1} - x^{k+1}),\end{aligned} \tag{6.4.15}$$

where the first equality is from (6.4.10), the second equality is from the inductive hypothesis, the third equality is from the iteration formula (6.3.2a), and the final equality is from the iteration formula (6.3.2b) with the definition of β_{k+1} from (6.4.1). We have thus proved (6.4.14), and by substituting this equality into (6.4.13),

we obtain that Φ^*_{k+1} is an upper bound on the right-hand side of (6.4.8). This establishes (6.4.6) and thus completes the proof of the theorem. □

6.5. Lower Bounds on Rates The term "optimal" in Nesterov's optimal method is used because the convergence rate achieved by the method is the best possible (possibly up to a constant), among algorithms that make use of gradient information at the iterates x^k. This claim can be proved by means of a carefully designed function, for which *no* method that makes use of all gradients observed up to and including iteration k (namely, $\nabla f(x^i)$, $i = 0, 1, 2, \ldots, k$) can produce a sequence $\{x^k\}$ that achieves a rate better than that of Theorem 6.3.5. The function proposed in [32] is a convex quadratic $f(x) = (1/2)x^T Ax - e_1^T x$, where

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 & \ldots & \ldots & 0 \\ -1 & 2 & -1 & 0 & \ldots & \ldots & 0 \\ 0 & -1 & 2 & -1 & 0 & \ldots & 0 \\ & & \ddots & \ddots & \ddots & & \\ 0 & \ldots & & 0 & -1 & 2 & -1 \\ 0 & \ldots & & & 0 & -1 & 2 \end{bmatrix}, \quad e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

The solution x^* satisfies $Ax^* = e_1$; its components are $x^*_i = 1 - i/(n+1)$, for $i = 1, 2, \ldots, n$. If we use $x^0 = 0$ as the starting point, and construct the iterate x^{k+1} as

$$x^{k+1} = x^k + \sum_{j=0}^{k} \xi_j \nabla f(x^j),$$

for some coefficients ξ_j, $j = 0, 1, \ldots, k$, an elementary inductive argument shows that each iterate x^k can have nonzero entries only in its first k components. It follows that for any such algorithm, we have

$$\|x^k - x^*\|^2 \geqslant \sum_{j=k+1}^{n} (x^*_j)^2 = \sum_{j=k+1}^{n} \left(1 - \frac{j}{n+1}\right)^2. \tag{6.5.1}$$

A little arithmetic shows that

$$\|x^k - x^*\|^2 \geqslant \frac{1}{8}\|x^0 - x^*\|^2, \quad k = 1, 2, \ldots, \frac{n}{2} - 1, \tag{6.5.2}$$

It can be shown further that

$$f(x^k) - f^* \geqslant \frac{3L}{32(k+1)^2}\|x^0 - x^*\|^2, \quad k = 1, 2, \ldots, \frac{n}{2} - 1, \tag{6.5.3}$$

where $L = \|A\|_2$. This lower bound on $f(x^k) - x^*$ is within a constant factor of the upper bound of Theorem 6.3.5.

The restriction $k \leqslant n/2$ in the argument above is not fully satisfying. A more compelling example would show that the lower bound (6.5.3) holds for *all* k, but an example of this type is not currently known.

7. Newton Methods

So far, we have dealt with methods that use first-order (gradient or subgradient) information about the objective function. We have shown that such algorithms can yield sequences of iterates that converge at linear or sublinear rates. We turn our attention in this chapter to methods that exploit second-derivative (Hessian) information. The canonical method here is Newton's method, named after Isaac Newton, who proposed a version of the method for polynomial equations in around 1670.

For many functions, including many that arise in data analysis, second-order information is not difficult to compute, in the sense that the functions that we deal with are simple (usually compositions of elementary functions). In comparing with first-order methods, there is a tradeoff. Second-order methods typically have local superlinear or quadratic convergence rates: Once the iterates reach a neighborhood of a solution at which second-order sufficient conditions are satisfied, convergence is rapid. Moreover, their global convergence properties are attractive. With appropriate enhancements, they can provably avoid convergence to saddle points. But the costs of calculating and handling the second-order information and of computing the step is higher. Whether this tradeoff makes them appealing depends on the specifics of the application and on whether the second-derivative computations are able to take advantage of structure in the objective function.

We start by sketching the basic Newton's method for the unconstrained smooth optimization problem $\min f(x)$, and prove local convergence to a minimizer x^* that satisfies second-order sufficient conditions. Subsection 7.2 discusses performance of Newton's method on convex functions, where the use of Newton search directions in the line search framework (4.0.1) can yield global convergence. Modifications of Newton's method for nonconvex functions are discussed in Subsection 7.3. Subsection 7.4 discusses algorithms for smooth nonconvex functions that use gradient and Hessian information but guarantee convergence to points that approximately satisfy second-order necessary conditions. Some variants of these methods are related closely to the trust-region methods discussed in Subsection 7.3, but the motivation and mechanics are somewhat different.

7.1. Basic Newton's Method Consider the problem

$$\min f(x), \tag{7.1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a Lipschitz twice continuously differentiable function, where the Hessian has Lipschitz constant M, that is,

$$\|\nabla^2 f(x') - \nabla^2 f(x'')\| \leqslant M\|x' - x''\|, \tag{7.1.2}$$

where $\|\cdot\|$ denotes the Euclidean vector norm and its induced matrix norm. Newton's method generates a sequence of iterates $\{x^k\}_{k=0,1,2,\ldots}$.

A second-order Taylor series approximation to f around the current iterate x^k is

$$f(x^k + p) \approx f(x^k) + \nabla f(x^k)^T p + \frac{1}{2} p^T \nabla^2 f(x^k) p. \tag{7.1.3}$$

When $\nabla^2 f(x^k)$ is positive definite, the minimizer p^k of the right-hand side is unique; it is

$$p^k = -\nabla^2 f(x^k)^{-1} \nabla f(x^k). \tag{7.1.4}$$

This is the Newton step. In its most basic form, then, Newton's method is defined by the following iteration:

$$x^{k+1} = x^k - \nabla^2 f(x^k)^{-1} \nabla f(x^k). \tag{7.1.5}$$

We have the following local convergence result in the neighborhood of a point x^* satisfying second-order sufficient conditions.

Theorem 7.1.6. *Consider the problem* (7.1.1) *with* f *twice Lipschitz continuously differentiable with Lipschitz constant* M *defined in* (7.1.2)*. Suppose that the second-order sufficient conditions are satisfied for the problem* (7.1.1) *at the point* x^**, that is,* $\nabla f(x^*) = 0$ *and* $\nabla^2 f(x^*) \succeq \gamma I$ *for some* $\gamma > 0$*. Then if* $\|x^0 - x^*\| \leqslant \frac{\gamma}{2M}$*, the sequence defined by* (7.1.5) *converges to* x^* *at a quadratic rate, with*

$$\|x^{k+1} - x^*\| \leqslant \frac{M}{\gamma} \|x^k - x^*\|^2, \quad k = 0, 1, 2, \ldots . \tag{7.1.7}$$

Proof. From (7.1.4) and (7.1.5), and using $\nabla f(x^*) = 0$, we have

$$\begin{aligned} x^{k+1} - x^* &= x^k - x^* - \nabla^2 f(x^k)^{-1} \nabla f(x^k) \\ &= \nabla^2 f(x^k)^{-1} [\nabla^2 f(x^k)(x^k - x^*) - (\nabla f(x^k) - \nabla f(x^*))]. \end{aligned}$$

so that

$$\|x^{k+1} - x^*\| \leqslant \|\nabla^2 f(x^k)^{-1}\| \|\nabla^2 f(x^k)(x^k - x^*) - (\nabla f(x^k) - \nabla f(x^*))\|. \tag{7.1.8}$$

By using Taylor's theorem (see (3.3.4) with $x = x^k$ and $p = x^* - x^k$), we have

$$\nabla f(x^k) - \nabla f(x^*) = \int_0^1 \nabla^2 f(x^k + t(x^* - x^k))(x^k - x^*)\, dt.$$

By using this result along with the Lipschitz condition (7.1.2), we have

$$\begin{aligned} &\|\nabla^2 f(x^k)(x^k - x^*) - (\nabla f(x^k) - \nabla f(x^*))\| \\ &= \left\| \int_0^1 [\nabla^2 f(x^k) - \nabla^2 f(x^k + t(x^* - x^k))](x^k - x^*)\, dt \right\| \\ &\leqslant \int_0^1 \|\nabla^2 f(x^k) - \nabla^2 f(x^k + t(x^* - x^k))\| \|x^k - x^*\|\, dt \\ &\leqslant \left(\int_0^1 Mt\, dt \right) \|x^k - x^*\|^2 = \tfrac{1}{2} M \|x^k - x^*\|^2. \end{aligned} \tag{7.1.9}$$

From the Weilandt-Hoffman inequality[28] and (7.1.2), we have that

$$|\lambda_{\min}(\nabla^2 f(x^k)) - \lambda_{\min}(\nabla^2 f(x^*))| \leqslant \|\nabla^2 f(x^k) - \nabla^2 f(x^*)\| \leqslant M \|x^k - x^*\|,$$

where $\lambda_{\min}(\cdot)$ denotes the smallest eigenvalue of a symmetric matrix. Thus for

$$\|x^k - x^*\| \leqslant \frac{\gamma}{2M}, \tag{7.1.10}$$

we have

$$\lambda_{\min}(\nabla^2 f(x^k)) \geqslant \lambda_{\min}(\nabla^2 f(x^*)) - M\|x^k - x^*\| \geqslant \gamma - M\frac{\gamma}{2M} \geqslant \frac{\gamma}{2},$$

so that $\|\nabla^2 f(x^k)^{-1}\| \leqslant 2/\gamma$. By substituting this result together with (7.1.9) into (7.1.8), we obtain

$$\|x^{k+1} - x^*\| \leqslant \frac{2}{\gamma}\frac{M}{2}\|x^k - x^*\|^2 = \frac{M}{\gamma}\|x^k - x^*\|^2,$$

verifying the local quadratic convergence rate. By applying (7.1.10) again, we have

$$\|x^{k+1} - x^*\| \leqslant \left(\frac{M}{\gamma}\|x^k - x^*\|\right)\|x^k - x^*\| \leqslant \frac{1}{2}\|x^k - x^*\|,$$

so, by arguing inductively, we see that the sequence converges to x^* provided that x^0 satisfies (7.1.10), as claimed. □

Of course, we do not need to explicitly identify a starting point x^0 in the stated region of convergence. Any sequence that approaches to x^* will eventually enter this region, and thereafter the quadratic convergence guarantees apply.

We have established that Newton's method converges rapidly once the iterates enter the neighborhood of a point x^* satisfying second-order sufficient optimality conditions. But what happens when we start far from such a point?

7.2. Newton's Method for Convex Functions When the function f is convex as well as smooth, we can devise variants of Newton's method for which global convergence and complexity results (in particular, results based on those of Section 4.5) can be proved in addition to local quadratic convergence.

When f is strongly convex with modulus γ and satisfies Lipschitz continuity of the gradient (3.3.6), the Hessian $\nabla^2 f(x^k)$ is positive definite for all k, with all eigenvalues in the interval $[\gamma, L]$. Thus, the Newton direction (7.1.4) is well defined at all iterates x^k, and is a descent direction satisfying the condition (4.5.1) with $\eta = \gamma/L$. To verify this claim, note first

$$\|p^k\| \leqslant \|\nabla^2 f(x^k)^{-1}\|\|\nabla f(x^k)\| \leqslant \frac{1}{\gamma}\|\nabla f(x^k)\|.$$

Then

$$\begin{aligned}(p^k)^T\nabla f(x^k) &= -\nabla f(x^k)^T\nabla^2 f(x^k)^{-1}\nabla f(x^k)\\ &\leqslant -\frac{1}{L}\|\nabla f(x^k)\|^2\\ &\leqslant -\frac{\gamma}{L}\|\nabla f(x^k)\|\|p^k\|.\end{aligned}$$

We can use the Newton direction in the line-search framework of Subsection 4.5 to obtain a method for which $x^k \to x^*$, where x^* is the (unique) global minimizer of f. (This claim follows from the property (4.5.6) together with the fact that x^* is the only point for which $\nabla f(x^*) = 0$.) We can even obtain a complexity result — and $O(1/\sqrt{T})$ bound on $\min_{0\leqslant k\leqslant T-1}\|\nabla f(x^k)\|$ — from Theorem 4.5.3.

These global convergence properties are enhanced by the local quadratic convergence property of Theorem 7.1.6 if we modify the line-search framework by accepting the step length $\alpha_k = 1$ in (4.0.1) whenever it satisfies the weak Wolfe conditions (4.5.2). (It can be shown, by again using arguments based on Taylor's theorem (Theorem 3.3.1), that these conditions *will* be satisfied by $\alpha_k = 1$ for all x^k sufficiently close to the minimizer x^*.)

Consider now the case in which f is convex and satisfies condition (3.3.6) but is not strongly convex. Here, the Hessian $\nabla^2 f(x^k)$ may be singular for some k, so the direction (7.1.4) may not be well defined. However, by adding any positive number $\lambda_k > 0$ to the diagonal, we can ensure that the modified Newton direction defined by

$$p^k = -[\nabla^2 f(x^k) + \lambda_k I]^{-1} \nabla f(x^k), \tag{7.2.1}$$

is well defined and is a descent direction for f. For any $\eta \in (0,1)$ in (4.5.1), we have by choosing λ_k large enough that $\lambda_k/(L+\lambda_k) \geqslant \eta$ that the condition (4.5.1) is satisfied too, so we can use the resulting direction p^k in the line-search framework of Subsection 4.5, to obtain a method that convergence to a solution x^* of (1.0.1), when one exists.

If, in addition, the minimizer x^* is unique and satisfies a second-order sufficient condition (so that $\nabla^2 f(x^*)$ is positive definite), then $\nabla^2 f(x^k)$ will be positive definite too for k sufficiently large. Thus, provided that η is sufficiently small, the *unmodified* Newton direction (with $\lambda_k = 0$ in (7.2.1)) will satisfy the condition (4.5.1). If we use (7.2.1) in the line-search framework of Section 4.5, but set $\lambda_k = 0$ where possible, and accept $\alpha_k = 1$ as the step length whenever it satisfies (4.5.2), we can obtain local quadratic convergence to x^*, in addition to the global convergence and complexity promised by Theorem 4.5.3.

7.3. Newton Methods for Nonconvex Functions For smooth nonconvex f, the Hessian $\nabla^2 f(x^k)$ may be indefinite for some k. The Newton direction (7.1.4) may not exist (when $\nabla^2 f(x^k)$ is singular) or it may not be a descent direction (when $\nabla^2 f(x^k)$ has negative eigenvalues). However, we can still define a modified Newton direction as in (7.2.1), which will be a descent direction for λ_k sufficiently large, and thus can be used in the line-search framework of Section 4.5. For a given η in (4.5.1), a sufficient condition for p^k from (7.2.1) to satisfy (4.5.1) is that

$$\frac{\lambda_k + \lambda_{\min}(\nabla^2 f(x^k))}{\lambda_k + L} \geqslant \eta,$$

where $\lambda_{\min}(\nabla^2 f(x^k))$ is the minimum eigenvalue of the Hessian, which may be negative. The line-search framework of Section 4.5 can then be applied to ensure that $\nabla f(x^k) \to 0$.

Once again, if the iterates $\{x^k\}$ enter the neighborhood of a local solution x^* for which $\nabla^2 f(x^*)$ is positive definite, some enhancements of the strategy for choosing λ_k and the step length α_k can recover the local quadratic convergence of Theorem 7.1.6.

Formula (7.2.1) is not the only way to modify the Newton direction to ensure descent in a line-search framework. Other approaches are outlined in [36, Chapter 3]. One such technique is to modify the Cholesky factorization of $\nabla^2 f(x^k)$ by adding positive elements to the diagonal only as needed to allow the factorization to proceed (that is, to avoid taking the square root of a negative number), then using the modified factorization in place of $\nabla^2 f(x^k)$ in the calculation of the Newton step p^k. Another technique is to compute an eigenvalue decomposition $\nabla^2 f(x^k) = Q_k \Lambda_k Q_k^T$ (where Q_k is orthogonal and Λ_k is the diagonal matrix containing the eigenvalues), then define $\tilde{\Lambda}_k$ to be a modified version of Λ^k in which all the diagonals are positive. Then, following (7.1.4), p^k can be defined as

$$p^k := -Q_k \tilde{\Lambda}_k^{-1} Q_k^T \nabla f(x^k).$$

When an appropriate strategy is used to define $\tilde{\Lambda}_k$, we can ensure satisfaction of the descent condition (4.5.1) for some $\eta > 0$. As above, the line-search framework of Section 4.5 can be used to obtain an algorithm that generates a sequence $\{x^k\}$ such that $\nabla f(x^k) \to 0$. We noted earlier that this condition ensures that all accumulation points $\hat{x}$ are stationary points, that is, they satisfy $\nabla f(\hat{x}) = 0$.

Stronger guarantees can be obtained from a *trust-region* version of Newton's method, which ensures convergence to a point satisfying second-order necessary conditions, that is, $\nabla^2 f(\hat{x}) \succeq 0$ in addition to $\nabla f(\hat{x}) = 0$. The trust-region approach was developed in the late 1970s and early 1980s, and has become popular again recently because of this appealing global convergence behavior. A trust-region Newton method also recovers quadratic convergence to solutions x^* satisfying second-order-sufficient conditions, without any special modifications. (The trust-region Newton approach is closely related to cubic regularization [26, 35], which we discuss in the next section.)

We now outline the trust-region approach. (Further details can be found in [36, Chapter 4].) The subproblem to be solved at each iteration is

$$\min_d \; f(x^k) + \nabla f(x^k)^T d + \frac{1}{2} d^T \nabla^2 f(x^k) d \quad \text{subject to } \|d\|_2 \leqslant \Delta_k. \tag{7.3.1}$$

The objective is a second-order Taylor-series approximation while Δ_k is the *radius* of the trust region — the region within which we trust the second-order model to capture the true behavior of f. Somewhat surprisingly, the problem (7.3.1) is not too difficult to solve, even when the Hessian $\nabla^2 f(x^k)$ is indefinite. In fact, the solution d^k of (7.3.1) satisfies the linear system

$$[\nabla^2 f(x^k) + \lambda I] d^k = -\nabla f(x^k), \quad \text{for some } \lambda \geqslant 0, \tag{7.3.2}$$

where λ is chosen such that $\nabla^2 f(x^k) + \lambda I$ is positive semidefinite and $\lambda > 0$ only if $\|d^k\| = \Delta_k$ (see [31]). Solving (7.3.1) thus reduces to a search for the appropriate value of the scalar λ_k, for which specialized methods have been devised.

For large-scale problems, it may be too expensive to solve (7.3.1) near-exactly, since the process may require several factorizations of an $n \times n$ matrix (namely, the coefficient matrix in (7.3.2), for different values of λ). A popular approach

for finding approximate solutions of (7.3.1), which can be used when $\nabla^2 f(x^k)$ is positive definite, is the *dogleg* method. In this method the curved path traced out by solutions of (7.3.2) for values of λ in the interval $[0, \infty)$ is approximated by simpler path consisting of two line segments. The first segment joins 0 to the point d_C^k that minimizes the objective in (7.3.1) along the direction $-\nabla f(x^k)$, while the second segment joins d_C^k to the pure Newton step defined in (7.1.4). The approximate solution is taken to be the point at which this "dogleg" path crosses the boundary of the trust region $\|d\| \leqslant \Delta_k$. If the dogleg path lies entirely inside the trust region, we take d^k to be the pure Newton step. See [36, Section 4.1].

Having discussed the trust-region subproblem (7.3.1), let us outline how it can be used as the basis for a complete algorithm. A crucial role is played by the ratio between the *amount of decrease in* f *predicted by the quadratic objective in* (7.3.1) and the *actual decrease in* f, *namely*, $f(x^k) - f(x^k + d^k)$. Ideally, this ratio would be close to 1. If it is at least greater than a small tolerance (say, 10^{-4}) we accept the step and proceed to the next iteration. Otherwise, we conclude that the trust-region radius Δ_k is too large, so we do not take the step, shrink the trust region, and re-solve (7.3.1) to obtain a new step. Additionally, when the actual-to-predicted ratio is close to 1, we conclude that a larger trust region may hasten progress, so we increase Δ for the next iteration, provided that the bound $\|d^k\| \leqslant \Delta_k$ really is active at the solution of (7.3.1).

Unlike a basic line-search method, the trust-region Newton method can "escape" from a saddle point. Suppose we have $\nabla f(x^k) = 0$ and $\nabla^2 f(x^k)$ indefinite with some strictly negative eigenvalues. Then, the solution d^k to (7.3.1) will be nonzero, and the algorithm will step away from the saddle point, in the direction of most negative curvature for $\nabla^2 f(x^k)$. Another appealing feature of the trust-region Newton approach is that when the sequence $\{x^k\}$ approaches a point x^* satisfying second-order sufficient conditions, the trust region bound becomes inactive, and the method takes pure Newton steps (7.1.4) for all sufficiently large k so the local quadratic convergence that characterizes Newton's method.

The basic difference between line-search and trust-region methods can be summarized as follows. Line-search methods first choose a direction p^k, then decide how far to move along that direction. Trust-region methods do the opposite: They choose the distance Δ_k first, then find the direction that makes the best progress for this step length.

7.4. A Cubic Regularization Approach Trust-region Newton methods have the significant advantage of guaranteeing that any accumulation points will satisfy second-order necessary conditions. A related approach based on *cubic regularization* has similar properties, plus some additional complexity guarantees. Cubic regularization requires the Hessian to be Lipschitz continuous, as in (7.1.2). It follows that the following cubic function yields a *global upper* bound for f:

$$T_M(z; x) := f(x) + \nabla f(x)^T (z - x) + \frac{1}{2}(z - x)^T \nabla^2 f(x)(z - x) + \frac{M}{6}\|z - x\|^3. \tag{7.4.1}$$

Specifically, we have for any x that

$$f(z) \leqslant T_M(z;x), \quad \text{for all } z.$$

The basic cubic regularization algorithm starting from x^0 proceeds as follows:

$$x^{k+1} = \arg\min_z T_M(z;x^k), \quad k = 0,1,2,\ldots. \tag{7.4.2}$$

The complexity properties of this approach were analyzed in [35], with variants being studied in [26] and [12,13]. Rather than present the theory for the method based on (7.4.2), we describe an elementary algorithm that makes use of the expansion (7.4.1) as well as the steepest-descent theory of Subsection 4.1. Our algorithm aims to identify a point that *approximately* satisfies second-order necessary conditions, that is,

$$\|\nabla f(x)\| \leqslant \epsilon_g, \quad \lambda_{\min}(\nabla^2 f(x)) \geqslant -\epsilon_H, \tag{7.4.3}$$

where ϵ_g and ϵ_H are two small constants. In addition to Lipschitz continuity of the Hessian (7.1.2), we assume Lipschitz continuity of the gradient with constant L (see (3.3.6)), and also that the objective f is lower-bounded by some number $\bar{f}$.

Our algorithm takes steps of two types: a steepest-descent step, as in Subsection 4.1, or a step in a negative curvature direction for $\nabla^2 f$. Iteration k proceeds as follows:

(i) If $\|\nabla f(x^k)\| > \epsilon_g$, take the steepest descent step (4.1.1).

(ii) Otherwise, if $\lambda_{\min}(\nabla^2 f(x^k)) < -\epsilon_H$, choose p^k to be the eigenvector corresponding to the most negative eigenvalue of $\nabla^2 f(x^k)$. Choose the size and sign of p^k such that $\|p^k\| = 1$ and $(p^k)^T \nabla f(x^k) \leqslant 0$, and set

$$x^{k+1} = x^k + \alpha_k p^k, \quad \text{where } \alpha_k = \frac{2\epsilon_H}{M}. \tag{7.4.4}$$

If neither of these conditions hold, then x^k satisfies the approximate second-order necessary conditions (7.4.3), so we terminate.

For the steepest-descent step (i), we have from (4.1.3) that

$$f(x^{k+1}) \leqslant f(x^k) - \frac{1}{2L}\|\nabla f(x^k)\|^2 \leqslant f(x^k) - \frac{\epsilon_g^2}{2L}. \tag{7.4.5}$$

For a step of type (ii), we have from (7.4.1) that

$$\begin{aligned} f(x^{k+1}) &\leqslant f(x^k) + \alpha_k \nabla f(x^k)^T p^k + \frac{1}{2}\alpha_k^2 (p^k)^T \nabla^2 f(x^k) p^k + \frac{1}{6} M \alpha_k^3 \|p^k\|^3 \\ &\leqslant f(x^k) - \frac{1}{2}\left(\frac{2\epsilon_H}{M}\right)^2 \epsilon_H + \frac{1}{6} M \left(\frac{2\epsilon_H}{M}\right)^3 \\ &= f(x^k) - \frac{2}{3}\frac{\epsilon_H^3}{M^2}. \end{aligned} \tag{7.4.6}$$

By aggregating (7.4.5) and (7.4.6), we have that at each x^k for which the condition (7.4.3) does *not* hold, we attain a decrease in the objective of at least

$$\min\left(\frac{\epsilon_g^2}{2L}, \frac{2}{3}\frac{\epsilon_H^3}{M^2}\right).$$

Using the lower bound $\bar{f}$ on the objective f, we see that the number of iterations K required must satisfy the condition

$$K \min\left(\frac{\epsilon_g^2}{2L}, \frac{2}{3}\frac{\epsilon_H^3}{M^2}\right) \leqslant f(x^0) - \bar{f},$$

from which we conclude that

$$K \leqslant \max\left(2L\epsilon_g^{-2}, \frac{3}{2}M^2\epsilon_H^{-3}\right)\left(f(x^0) - \bar{f}\right).$$

We also observe that that the maximum number of iterates required to identify a point at which only the approximate stationarity condition $\|\nabla f(x^k)\| \leqslant \epsilon_g$ holds is $2L\epsilon_g^{-2}(f(x^0) - \bar{f})$. (We can just omit the second-order part of the algorithm.) Note too that it is easy to devise *approximate* versions of this algorithm with similar complexity. For example, the negative curvature direction p^k in step (ii) above can be replaced by an approximation to the direction of most negative curvature, obtained by the Lanczos iteration with random initialization.

In algorithms that make more complete use of the cubic model (7.4.1), the term ϵ_g^{-2} in the complexity expression becomes $\epsilon_g^{-3/2}$, and the constants are different. The subproblems (7.4.1) are more complicated to solve than those in the simple scheme above. Active research is going on into other algorithms that achieve complexities similar to those of the cubic regularization approach. A variety of methods that make use of Newton-type steps, approximate negative curvature directions, accelerated gradient methods, random perturbations, randomized Lanczos and conjugate gradient methods, and other algorithmic elements have been proposed.

8. Conclusions

We have outlined various algorithmic tools from optimization that are useful for solving problems in data analysis and machine learning, and presented their basic theoretical properties. The intersection of optimization and machine learning is a fruitful and very popular area of current research. All the major machine learning conferences have a large contingent of optimization papers, and there is a great deal of interest in developing algorithmic tools to meet new challenges and in understanding their properties. The edited volume [41] contains a snapshot of the state of the art circa 2010, but this is a fast-moving field and there have been many developments since then.

Acknowledgments

I thank Ching-pei Lee for a close reading and many helpful suggestions, and David Hong and an anonymous referee for detailed, excellent comments.

References

[1] L. Balzano, R. Nowak, and B. Recht, *Online identification and tracking of subspaces from highly incomplete information*, 48th Annual Allerton Conference on Communication, Control, and Computing, 2010, pp. 704–711. ←9

[2] L. Balzano and S. J. Wright, *Local convergence of an algorithm for subspace identification from partial data*, Foundations of Computational Mathematics **14** (2014), 1–36. DOI: 10.1007/s10208-014-9227-7. ←10

[3] A. Beck and M. Teboulle, *A fast iterative shrinkage-threshold algorithm for linear inverse problems*, SIAM Journal on Imaging Sciences **2** (2009), no. 1, 183–202. ←32, 35

[4] B. E. Boser, I. M. Guyon, and V. N. Vapnik, *A training algorithm for optimal margin classifiers*, Proceedings of the Fifth Annual Workshop on Computational Learning Theory, 1992, pp. 144–152. ←12

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction methods of multipliers*, Foundations and Trends in Machine Learning **3** (2011), no. 1, 1–122. ←3

[6] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge University Press, 2003. ←3

[7] S. Bubeck, *Convex optimization: Algorithms and complexity*, Foundations and Trends in Machine Learning **8** (2015), no. 3–4, 231–357. ←35, 36

[8] S. Bubeck, Y. T. Lee, and M. Singh, *A geometric alternative to Nesterov's accelerated gradient descent*, Technical Report arXiv:1506.08187, Microsoft Research, 2015. ←35

[9] S. Burer and R. D. C. Monteiro, *A nonlinear programming algorithm for solving semidefinite programs via low-rank factorizations*, Mathematical Programming, Series B **95** (2003), 329–257. ←7

[10] E. Candès and B. Recht, *Exact matrix completion via convex optimization*, Foundations of Computational Mathematics **9** (2009), 717–772. ←7

[11] E. J. Candès, X. Li, Y. Ma, and J. Wright, *Robust principal component analysis?*, Journal of the ACM **58.3** (2011), 11. ←9

[12] C. Cartis, N. I. M. Gould, and Ph. L. Toint, *Adaptive cubic regularisation methods for unconstrained optimization. Part I: Motivation, convergence and numerical results*, Mathematical Programming, Series A **127** (2011), 245–295. ←46

[13] C. Cartis, N. I. M. Gould, and Ph. L. Toint, *Adaptive cubic regularisation methods for unconstrained optimization. Part II: Worst-case function-and derivative-evaluation complexity*, Mathematical Programming, Series A **130** (2011), no. 2, 295–319. ←46

[14] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, *Rank-sparsity incoherence for matrix decomposition*, SIAM Journal on Optimization **21** (2011), no. 2, 572–596. ←9

[15] Y. Chen and M. J. Wainwright, *Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees*, Technical Report arXiv:1509.03025, University of California-Berkeley, 2015. ←9

[16] C. Cortes and V. N. Vapnik, *Support-vector networks*, Machine Learning **20** (1995), 273–297. ←12

[17] A. d'Aspremont, O. Banerjee, and L. El Ghaoui, *First-order methods for sparse covariance selection*, SIAM Journal on Matrix Analysis and Applications **30** (2008), 56–66. ←8

[18] A. d'Aspremont, L. El Ghaoui, M. I. Jordan, and G. Lanckriet, *A direct formulation for sparse PCA using semidefinte programming*, SIAM Review **49** (2007), no. 3, 434–448. ←8

[19] T. Dasu and T. Johnson, *Exploratory data mining and data cleaning*, John Wiley & Sons, 2003. ←4

[20] P. Drineas and M. W. Mahoney, *Lectures on randomized numerical linear algebra*, The mathematics of data, 2018. ←6

[21] D. Drusvyatskiy, M. Fazel, and S. Roy, *An optimal first-order method based on optimal quadratic averaging*, Technical Report arXiv:1604.06543, University of Washington, 2016. To appear in SIAM Journal on Optimization. ←35

[22] J. C. Duchi, *Introductory lectures on stochastic optimization*, The mathematics of data, 2018. ←3, 16, 23

[23] J. Eckstein and D. P. Bertsekas, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Mathematical Programming **55** (1992), 293–318. ←3

[24] M. Frank and P. Wolfe, *An algorithm for quadratic programming*, Naval Research Logistics Quarterly **3** (1956), 95–110. ←28

[25] J. Friedman, T. Hastie, and R. Tibshirani, *Sparse inverse covariance estimation with the graphical lasso*, Biostatistics **9** (2008), no. 3, 432–441. ←8

[26] A. Griewank, *The modification of Newton's method for unconstrained optimization by bounding cubic terms*, Technical Report NA/12, DAMTP, Cambridge University, 1981. ←44, 46
[27] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, Journal of Research of the National Bureau of Standards **49** (1952), 409–436. ←33
[28] A. J. Hoffman and H. Weilandt, *The variation of the spectrum of a normal matrix*, Duke Mathematical Journal **20** (1953), no. 1, 37–39. ←41
[29] J. D Lee, M. Simchowitz, M. I Jordan, and B. Recht, *Gradient descent only converges to minimizers*, Conference on learning theory, 2016, pp. 1246–1257. ←27
[30] D. C. Liu and J. Nocedal, *On the limited-memory BFGS method for large scale optimization*, Mathematical Programming **45** (1989), 503–528. ←3, 34
[31] J. J. Moré and D. C. Sorensen, *Computing a trust region step*, SIAM Journal on Scientific and Statistical Computing **4** (1983), 553–572. ←44
[32] A. S. Nemirovski and D. B. Yudin, *Problem complexity and method efficiency in optimization*, John Wiley, 1983. ←39
[33] Y. Nesterov, *A method for unconstrained convex problem with the rate of convergence* $O(1/k^2)$, Doklady AN SSSR **269** (1983), 543–547. ←32
[34] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*, Springer Science and Business Media, New York, 2004. ←32, 36
[35] Y. Nesterov and B. T. Polyak, *Cubic regularization of Newton method and its global performance*, Mathematical Programming, Series A **108** (2006), 177–205. ←44, 46
[36] J. Nocedal and S. J. Wright, *Numerical Optimization*, Second, Springer, New York, 2006. ←3, 26, 34, 44, 45
[37] B. T. Polyak, *Some methods of speeding up the convergence of iteration methods*, USSR Computational Mathematics and Mathematical Physics **4.5** (1964), 1–17. ←32
[38] B. T. Polyak, *Introduction to optimization*, Optimization Software, 1987. ←32, 33
[39] B. Recht, M. Fazel, and P. Parrilo, *Guaranteed minimum-rank solutions to linear matrix equations via nuclear norm minimization*, SIAM Review **52** (2010), no. 3, 471–501. ←7
[40] R. T. Rockafellar, *Convex analysis*, Princeton University Press, Princeton, N.J., 1970. ←17
[41] S. Sra, S. Nowozin, and S. J. Wright (eds.), *Optimization for machine learning*, NIPS Workshop Series, MIT Press, 2011. ←47
[42] R. Tibshirani, *Regression shrinkage and selection via the LASSO*, Journal of the Royal Statistical Society B **58** (1996), 267–288. ←6
[43] M. J. Todd, *Semidefinite optimization*, Acta Numerica **10** (2001), 515–560. ←3
[44] B. Turlach, W. N. Venables, and S. J. Wright, *Simultaneous variable selection*, Technometrics **47** (2005), no. 3, 349–363. ←9
[45] L. Vandenberghe and S. Boyd, *Semidefinite programming*, SIAM Review **38** (1996), 49–95. ←3
[46] S. J. Wright, *Primal-dual interior-point methods*, SIAM, Philadelphia, PA, 1997. ←3
[47] S. J. Wright, *Coordinate descent algorithms*, Mathematical Programming, Series B **151** (2015December), 3–34. ←3
[48] X. Yi, D. Park, Y. Chen, and C. Caramanis, *Fast algorithms for robust PCA via gradient descent*, Advances in Neural Information Processing Systems 29, 2016, pp. 4152–4160. ←9

Computer Sciences Department, University of Wisconsin-Madison, Madison, WI 53706
E-mail address: swright@cs.wisc.edu