11 Projected Newton-type Methods in Machine Learning

Mark Schmidt schmidtmarkw@gmail.com
University of British Columbia
Vancouver, BC, V6T 1Z4

Dongmin Kim dmkim@cs.utexas.edu
University of Texas at Austin
Austin, Texas 78712

Suvrit Sra suvrit@tuebingen.mpg.de
Max Planck Institute for Intelligent Systems
72076, Tübingen, Germany

We consider projected Newton-type methods for solving large-scale optimization problems arising in machine learning and related fields. We first introduce an algorithmic framework for projected Newton-type methods by reviewing a canonical projected (quasi-)Newton method. This method, while conceptually pleasing, has a high computation cost per iteration. Thus, we discuss two variants that are more scalable: two-metric projection and inexact projection methods. Finally, we show how to apply the Newton-type framework to handle nonsmooth objectives. Examples are provided throughout the chapter to illustrate machine learning applications of our framework.

11.1 Introduction

We study Newton-type methods for solving the optimization problem

$$\min_{\boldsymbol{x}} \quad f(\boldsymbol{x}) + r(\boldsymbol{x}), \quad \text{subject to} \quad \boldsymbol{x} \in \Omega, \tag{11.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable and convex; $r : \mathbb{R}^n \to \mathbb{R}$ is continuous and convex, but not necessarily differentiable everywhere; and Ω is a simple convex constraint set. This formulation is general and captures numerous problems in machine learning, especially where f corresponds to a loss, and r to a regularizer. Let us, however, defer concrete examples of (11.1) until we have developed some theoretical background.

We propose to solve (11.1) via Newton-type methods, a certain class of second-order methods that are known to often work well for unconstrained problems. For constrained problems too, we may consider Newton-type methods that, akin to their unconstrained versions, iteratively minimize a quadratic approximation to the objective, this time subject to constraints. This idea dates back to Levitin and Polyak (1966, §7), and it is referred to as a *projected Newton* method.

Projected Newton methods for optimization over convex sets share many of the appealing properties of their unconstrained counterparts. For example, their iterations are guaranteed to improve the objective function for a small enough stepsize; global convergence can be shown under a variant of the Armijo condition; and rapid local convergence rates can be shown around local minima satisfying strong convexity (Bertsekas, 1999). In a similar vein, we may consider projected quasi-Newton methods, where we interpolate differences in parameter and gradient values to approximate the Hessian matrix. The resulting *Newton-type* methods are the subject of this chapter, and we will focus particularly on the limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) quasi-Newton approximation. The main appeals of the L-BFGS approximation are its linear time iteration complexity and its strong empirical performance on a variety of problems.

The remainder of this chapter is organized as follows. We first restrict ourselves to smooth optimization, where $r(\boldsymbol{x}) = 0$. For this setting, we describe projected Newton-type methods (Section 11.2), covering basic implementation issues such as Hessian approximation and line-search. Then, we describe two-metric projection methods (Section 11.3), followed by inexact (or truncated) projected-Newton methods (Section 11.4). Finally, we discuss the nonsmooth setting, where $r(\boldsymbol{x}) \neq 0$, for which we describe two Newton-type methods (Section 11.5).

11.2 Projected Newton-type Methods

Projected Newton-type methods optimize their objective iteratively. At iteration k, they first approximate the objective function around the current iterate $\boldsymbol{x}^k$ by the quadratic model

$$\mathcal{Q}^k(\boldsymbol{x},\alpha) \triangleq f(\boldsymbol{x}^k) + (\boldsymbol{x}-\boldsymbol{x}^k)^T\nabla f(\boldsymbol{x}^k) + \frac{1}{2\alpha}(\boldsymbol{x}-\boldsymbol{x}^k)^T\boldsymbol{H}^k(\boldsymbol{x}-\boldsymbol{x}^k). \quad (11.2)$$

This model is parameterized by a positive stepsize α, and it uses a positive definite matrix $\boldsymbol{H}^k$ to approximate the Hessian $\nabla^2 f(\boldsymbol{x}^k)$. To generate the next iterate that decreases the objective while remaining feasible, the methods minimize the quadratic model (11.2) over the (*convex*) constraint set Ω. Thus, for a fixed $\alpha > 0$, they compute the unique element

$$\bar{\boldsymbol{x}}^k_\alpha = \underset{\boldsymbol{x}\in\Omega}{\operatorname{argmin}} \quad \mathcal{Q}^k(\boldsymbol{x},\alpha), \quad (11.3)$$

which is then used to obtain the new iterate by simply setting

$$\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k + \beta(\bar{\boldsymbol{x}}^k_\alpha - \boldsymbol{x}^k), \quad (11.4)$$

where $\beta \in (0,1]$ is another stepsize. To ensure a sufficient decrease in the objective value, one typically begins by setting $\alpha = \beta = 1$, and then decreases one of them until $\boldsymbol{x}^{k+1}$ satisfies the following Armijo condition[1]

$$f(\boldsymbol{x}^{k+1}) \le f(\boldsymbol{x}^k) + \nu\langle\nabla f(\boldsymbol{x}^k),\, \boldsymbol{x}^{k+1} - \boldsymbol{x}^k\rangle, \qquad \nu \in (0,1). \quad (11.5)$$

We collect the above described steps into Algorithm 11.1, which we present as the general framework for projected Newton-type methods.

Algorithm 11.1 A projected Newton-type method.

```
Given $\boldsymbol{x}^0 \in \Omega$, $\boldsymbol{H}^0 \succ \boldsymbol{0}$
for $k = 0, \ldots,$ until some stopping criteria met do
    Step I: Build $\mathcal{Q}^k(\boldsymbol{x},\alpha)$ using (11.2)
    repeat
        Step IIa: Minimize $\mathcal{Q}^k(\boldsymbol{x},\alpha)$ over $\Omega$
        Step IIb: Update $\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k + \beta(\bar{\boldsymbol{x}}^k_\alpha - \boldsymbol{x}^k)$
        Step III: Update $\alpha$ and/or $\beta$
    until descent condition (11.5) is satisfied
end for
```

Convergence properties of various forms of this method are discussed, for example, in Bertsekas (1999, Section 2.3). In particular, convergence to a stationary point can be shown under the assumption that the eigenvalues of $\boldsymbol{H}^k$ are bounded between two positive constants. Also, if $\boldsymbol{x}^*$ is a minimizer of $f(\boldsymbol{x})$ over Ω satisfying certain conditions, and once $\boldsymbol{x}^k$ is sufficiently close

1. A typical value for the sufficient decrease parameter ν is 10^{-4}.

to $\boldsymbol{x}^*$, then $\alpha = 1$ and $\beta = 1$ are accepted as stepsizes and the sequence $||\boldsymbol{x}^k - \boldsymbol{x}^*||$ converges to zero at a superlinear rate.

Algorithm 11.1 is conceptually simple, and thus appealing. It can, however, have numerous variants, depending on how each step is implemented. For example, in Step I, which particular quadratic model is used; in Step II, how we minimize the model function; and in Step III, how we compute the stepsizes α and β. For each of these three steps there are multiple possible choices, and consequently, different combinations lead to methods of differing character. We describe some popular implementation choices below.

11.2.1 Building a Quadratic Model

When a positive definite Hessian is readily available, we can simply set $\boldsymbol{H}^k = \nabla^2 f(\boldsymbol{x}^k)$. By doing so, the quadratic model (11.2) becomes merely the approximation obtained via a second-order Taylor expansion of f. This model leads to computation of an *exact* Newton step at each iteration. At the other extreme, if we select $\boldsymbol{H}^k = \boldsymbol{I}$, the identity matrix of appropriate size, then the search direction of the resulting method reduces to the negative gradient, essentially yielding the projected gradient method. These two strategies often contrast with each other in terms of computing a search (descent) direction: the Newton step is considered one of the most sophisticated, while the gradient step is regarded as one of the simplest. In cases where we can efficiently compute the Euclidean projection operator, projected gradient steps have a low per-iteration computational cost. However, this benefit comes at the expense of linear convergence speed. The Newton step is usually more expensive; Step IIb will typically be costly to solve even if we can efficiently compute the Euclidean projection onto the constraint set. However, the more expensive Newton step generally enjoys a local superlinear convergence rate.

Despite its theoretical advantages, an exact Newton step often is resource intensive, especially when computing the exact Hessian is expensive. To circumvent some of the associated computational issues, one usually approximates the Hessian: this idea underlies the well-known quasi-Newton approximation. Let us therefore briefly revisit the BFGS update that approximates the exact Hessian.

11.2.1.1 BFGS *Update*

There exist several approximations to the Hessian, such as the Powell-Symmetric-Broyden (PSB), the Davidson-Fletcher-Powell (DFP), and the Broyden-Fletcher-Goldfarb-Shanno (BFGS). We focus on BFGS because it is believed to be the most effective in general (Gill et al., 1981; Bertsekas, 1999).

First, define the difference vectors $\boldsymbol{g}$ and $\boldsymbol{s}$ as follows:

$$\boldsymbol{g} = \nabla f(\boldsymbol{x}^{k+1}) - \nabla f(\boldsymbol{x}^k), \quad \text{and} \quad \boldsymbol{s} = \boldsymbol{x}^{k+1} - \boldsymbol{x}^k.$$

Now, assume we already have $\boldsymbol{H}^k$, the current approximation to the Hessian. Then, the BFGS update adds a rank-two correction to $\boldsymbol{H}^k$ to obtain

$$\boldsymbol{H}^{k+1} = \boldsymbol{H}^k - \frac{\boldsymbol{H}^k \boldsymbol{s}\boldsymbol{s}^T \boldsymbol{H}^k}{\boldsymbol{s}^T \boldsymbol{H}^k \boldsymbol{s}} + \frac{\boldsymbol{g}\boldsymbol{g}^T}{\boldsymbol{s}^T \boldsymbol{g}}. \tag{11.6}$$

We can plug $\boldsymbol{H}^{k+1}$ into (11.2) to obtain an updated model $\mathcal{Q}^{k+1}$. But depending on the implementation of subsequent steps (11.3) and (11.4), it might be more convenient and computationally efficient to update an estimate to the inverse of $\boldsymbol{H}^k$ instead. For this case, we can apply the Sherman-Morrison-Woodbury formula to (11.6), thus obtaining the update

$$\boldsymbol{S}^{k+1} = \boldsymbol{S}^k + \left(1 + \frac{\boldsymbol{g}^T \boldsymbol{S}^k \boldsymbol{g}}{\boldsymbol{s}^T \boldsymbol{g}}\right) \frac{\boldsymbol{s}\boldsymbol{s}^T}{\boldsymbol{s}^T \boldsymbol{g}} - \frac{(\boldsymbol{S}^k \boldsymbol{g}\boldsymbol{s}^T + \boldsymbol{s}\boldsymbol{g}^T \boldsymbol{S}^k)}{\boldsymbol{s}^T \boldsymbol{g}}, \tag{11.7}$$

where $\boldsymbol{S}^k$ is the inverse of $\boldsymbol{H}^k$, also known as the *gradient scaling* matrix.

11.2.1.2 Limited Memory BFGS *Update.*

Though the BFGS update may greatly relieve the burden of Hessian computation, it still requires the same storage: $\mathcal{O}(n^2)$ for dense $\boldsymbol{H}^k$ or $\boldsymbol{S}^k$, which is troublesome for large-scale problems. This difficulty is addressed by the limited memory BFGS (L-BFGS) update, where, instead of using full matrices $\boldsymbol{H}^k$ and $\boldsymbol{S}^k$, a small number of vectors, say m, are used to approximate the Hessian or its inverse. The standard L-BFGS approach (Nocedal, 1980) can be implemented using the following formula (Nocedal and Wright, 2000)

$$\begin{aligned} \boldsymbol{S}^k = & \frac{\boldsymbol{s}_{k-1}^T \boldsymbol{g}_{k-1}}{\boldsymbol{g}_{k-1}^T \boldsymbol{g}_{k-1}} \bar{\boldsymbol{V}}_{k-M}^T \bar{\boldsymbol{V}}_{k-M} + \rho_{k-M} \bar{\boldsymbol{V}}_{k-M+1}^T \boldsymbol{s}_{k-M} \boldsymbol{s}_{k-M}^T \bar{\boldsymbol{V}}_{k-M+1} \\ & + \rho_{k-M+1} \bar{\boldsymbol{V}}_{k-M+2}^T \boldsymbol{s}_{k-M+1} \boldsymbol{s}_{k-M+1}^T \bar{\boldsymbol{V}}_{k-M+2} \\ & + \cdots \\ & + \rho_{k-1} \boldsymbol{s}_{k-1} \boldsymbol{s}_{k-1}^T, \end{aligned} \tag{11.8}$$

for $k \geq 1$; the scalars ρ_k and matrices $\bar{\boldsymbol{V}}_{k-M}$ are defined by

$$\rho_k = 1/(\boldsymbol{s}_k^T \boldsymbol{g}_k), \qquad \bar{\boldsymbol{V}}_{k-M} = [\boldsymbol{V}_{k-M} \cdots \boldsymbol{V}_{k-1}], \quad \text{and} \quad \boldsymbol{V}_k = I - \rho_k \boldsymbol{s}_k \boldsymbol{g}_k^T.$$

The L-BFGS approximation requires only $\mathcal{O}(mn)$ storage; moreover, multiplication of $\boldsymbol{S}^k$ by a vector can also be performed at this cost.

For both BFGS and L-BFGS, a choice that can significantly impact performance is the initial approximation $\boldsymbol{H}^0$. A typical strategy to select $\boldsymbol{H}^0$ is to set it to the negative gradient direction on the first iteration, and then to set $\boldsymbol{H}^0 = (\boldsymbol{g}^T\boldsymbol{g})/(\boldsymbol{g}^T\boldsymbol{s})\boldsymbol{I}$ on the next iteration. This choice was proposed by Shanno and Phua (1978) to optimize the condition number of the approximation. In the L-BFGS method we can reset $\boldsymbol{H}^0$ using this formula after *each* iteration (Nocedal and Wright, 2000, Section 7.2). With this strategy, the unit stepsizes of $\alpha = 1$ and $\beta = 1$ are typically accepted, which may remove the need for a line search on most iterations.

Provided that $\boldsymbol{H}^0$ is positive definite, the subsequent (implicit) Hessian approximations $\boldsymbol{H}^k$ generated by the L-BFGS update are guaranteed to be positive definite as long as $\boldsymbol{g}^T\boldsymbol{s}$ is positive (Nocedal and Wright, 2000, Section 6.1). This positivity is guaranteed if $f(\boldsymbol{x})$ is strongly convex, but when $f(\boldsymbol{x})$ is not strongly convex a more advanced strategy is required, see for instance Nocedal and Wright (2000, Section 18.3).

11.2.2 Solving the Subproblem

With our quadratic approximation $\mathcal{Q}^k(\boldsymbol{x}, \alpha)$ in hand, the next step is to solve the subproblem (11.3). For $\alpha \neq 0$, simple rearrangement shows that[2]

$$\bar{\boldsymbol{x}}^k_\alpha \;=\; \underset{\boldsymbol{x}\in\Omega}{\operatorname{argmin}}\; \mathcal{Q}^k(\boldsymbol{x}, \alpha) \;=\; \underset{\boldsymbol{x}\in\Omega}{\operatorname{argmin}}\; \frac{1}{2}||\boldsymbol{x} - \boldsymbol{y}^k||^2_{\boldsymbol{H}^k}, \tag{11.9}$$

where $\|\boldsymbol{x}\|_{\boldsymbol{H}^k}$ is defined by the norm $\sqrt{\boldsymbol{x}^T\boldsymbol{H}^k\boldsymbol{x}}$, and $\boldsymbol{y}^k$ is the unconstrained Newton step: $\boldsymbol{y}^k = \boldsymbol{x}^k - \alpha[\boldsymbol{H}^k]^{-1}\nabla f(\boldsymbol{x}^k)$. In words, $\bar{\boldsymbol{x}}^k_\alpha$ is obtained by projecting the Newton step onto the constraint set Ω, where projection is with respect to the metric defined by the Hessian approximation $\boldsymbol{H}^k$.

One major drawback of (11.9) is that it can be computationally challenging, even when Ω has relatively simple structure. To ease the computational burden, instead of using the metric defined by $\boldsymbol{H}^k$, we could compute the projection under the standard Euclidean norm while slightly modifying the Newton step to ensure convergence. This is the subject of Section 11.3. Alternatively, in Section 11.4 we consider computing an *approximate* solution to (11.9) itself.

Note that if we replace $\boldsymbol{H}^k$ with $\boldsymbol{I}$, both as the projection metric and in

2. If we use $\boldsymbol{S}^k$, the inverse of the Hessian, then $\bar{\boldsymbol{x}}^k_\alpha$ may be equivalently obtained by solving $\operatorname{argmin}_{\boldsymbol{x}\in\Omega} \frac{1}{2}||\boldsymbol{x} - (\boldsymbol{x}^k - \alpha\boldsymbol{S}^k\nabla f(\boldsymbol{x}^k))||^2_{[\boldsymbol{S}^k]^{-1}} = \operatorname{argmin}_{\boldsymbol{x}\in\Omega} \frac{1}{2}||\boldsymbol{x} - \boldsymbol{y}^k||^2_{[\boldsymbol{S}^k]^{-1}}$.

the Newton step, we recover gradient projection methods.

11.2.3 Computing the Stepsizes

Consider the stepsizes α and β in (11.3) and (11.4). Generally speaking, any positive α and β that generate $\boldsymbol{x}^{k+1}$ satisfying the descent condition (11.5) are acceptable. Practical choices are discussed below.

11.2.3.1 Backtracking

Suppose we fix $\alpha = 1$ for all k, and let $\boldsymbol{d}^k = \bar{\boldsymbol{x}}_1^k - \boldsymbol{x}^k$. Then we obtain the following update:

$$\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k + \beta \boldsymbol{d}^k.$$

To select β, we can simply start with $\beta = 1$ and iteratively decrease β until the resulting $\boldsymbol{x}^{k+1}$ satisfies (11.5)[3]. More formally, we set $\beta = \tau \cdot \sigma^m$ for some $\tau > 0$ and $\sigma \in (0, 1)$, where $m \geq 0$ is the first integer that satisfies

$$f(\boldsymbol{x}^{k+1}) \leq f(\boldsymbol{x}^k) + \tau \cdot \sigma^m \nabla f(\boldsymbol{x}^k)(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k).$$

Several strategies are available to reduce the number of backtracking iterations. For example, rather than simply dividing the stepsize by a constant, we can use information collected about the function during the line search to make a more intelligent choice. For example, if some trial value of β is not accepted, then we can set the next β to the minimum of the quadratic polynomial that has a value of $f(\boldsymbol{x}^k)$ at zero, $f(\boldsymbol{x}^k + \beta \boldsymbol{d}^k)$ at β, and a slope of $\nabla f(\boldsymbol{x}^k)^T \boldsymbol{d}^k$ at zero (Nocedal and Wright, 2000, Section 3.5). This choice gives the optimal stepsize if $f(\boldsymbol{x})$ is a quadratic function, and often drastically reduces the number of backtracking iterations needed. For some functions, quadratic interpolation can also be used to give a more intelligent choice than $\beta = 1$ for the *first* trial value of β, while cubic interpolation can be used if we have tested more than one value of β or if we compute $\nabla f(\boldsymbol{x}^k + \beta \boldsymbol{d}^k)$ for the trial values of β (Nocedal and Wright, 2000, Section 3.5).

11.2.3.2 Backtracking (Armijo) along projection arc (Bertsekas, 1999)

Alternatively, we can set $\beta = 1$ for all k to obtain

$$\boldsymbol{x}^{k+1} \leftarrow \bar{\boldsymbol{x}}_\alpha^k,$$

3. Also known as Armijo backtracking along a feasible direction.

and then determine an α satisfying (11.5). Similar to simple backtracking, we compute $\alpha = s \cdot \sigma^m$ for some $s, \tau > 0$, and $\sigma \in (0, 1)$, where $m \geq 0$ is the first integer that satisfies

$$f(\bar{\boldsymbol{x}}^k_{s\cdot\sigma^m}) \leq f(\boldsymbol{x}^k) + \tau \nabla f(\boldsymbol{x}^k)(\bar{\boldsymbol{x}}^k_{s\cdot\sigma^m} - \boldsymbol{x}^k).$$

Unlike simple backtracking that searches along a line segment as β varies, this strategy searches along a potentially nonlinear path as α varies. Because of this, polynomial interpolation to select trial values of α is more tenuous than for simple backtracking, but on many problems polynomial interpolation still significantly decreases the number of trial values evaluated.

This stepsize computation might be more involved when computing projections onto Ω is expensive, since it requires solving an optimization problem to compute $\bar{\boldsymbol{x}}^k_\alpha$ for each trial value of α. However, it can still be appealing because it is more likely to yield iterates that lie on the boundaries of the constraints. This property is especially useful when the boundaries of the constraints represent a solution of interest, such as *sparse* solutions with the constraint set $\Omega = \mathbb{R}^n_+$.

We next consider a few specific instantiations of the general framework introduced in this section. Specifically, we first consider two-metric projection methods for the specific case of bound-constrained problems (Section 11.3). Subsequently, we consider inexact projected Newton methods for optimization over more general simple convex sets (Section 11.4). Finally, we explore the versatility of the framework by extending it to problems with nonsmooth objective functions (Section 11.5).

11.3 Two-Metric Projection Methods

As mentioned earlier, computing the projection with respect to a quadratic norm defined by $\boldsymbol{H}^k$ can be computationally challenging. However, we often encounter problems with simple convex domains, onto which we can efficiently compute Euclidean projections. For optimization over such domains, we might therefore prefer projecting the Newton step under the Euclidean norm. Indeed, this choice is made by the well-known *two-metric projection* method, so named because it uses different matrices (metrics) for scaling the gradient and for computing the projection.

In two-metric projection algorithms, we can benefit from low iteration complexity if we use L-BFGS approximations. However, some problems still persist: the "obvious" procedure with an unmodified Newton step may not improve on the objective function, even for an arbitrarily small positive stepsize. Nevertheless, there are many cases where one can derive a two-metric projection method that can dodge this drawback without giving up the attractive properties of its unconstrained counterpart. A particular example is Bertsekas's projected Newton method (Bertsekas, 1982), and we discuss it below for the case where Ω consists of bound constraints.

The projected-Newton method may be viewed in light of Algorithm 11.1. Specifically, it takes the Hessian $\nabla^2 f(\boldsymbol{x}^k)$ and modifies its *inverse* so that the gradient scaling matrix $\boldsymbol{S}^k$ has a special structure. It subsequently invokes orthogonal projection in Step II, and then in Step III, it computes its stepsize using backtracking along the projection arc. The key variation from Algorithm 11.1 lies in how to modify the inverse Hessian to obtain a valid gradient scaling; the details follow below.

11.3.1 Bound Constrained Smooth Convex Problems

Consider the following special case of (11.1)

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \quad f(\boldsymbol{x}), \quad \text{subject to} \quad \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}, \tag{11.10}$$

where $\boldsymbol{l}$ and $\boldsymbol{u}$ are fixed vectors, and inequalities are taken componentwise (which can be set to ∞ or $-\infty$ if the variables are unbounded). The function f is assumed to be convex and twice continuously differentiable. Such bound-constrained problems arise as Lagrangian duals of problems with convex inequality constraints, or when we have natural restrictions (e.g., nonnegativity) on the variables. For bound-constrained problems the projection under the Euclidean norm is the standard orthogonal projection obtained by taking componentwise medians among l_i,u_i, and x_i:

$$[\mathcal{P}(\boldsymbol{x})]_i \triangleq \operatorname{mid}\{l_i, x_i, u_i\}.$$

At each iteration, we partition the variables into two groups: *free* and *restricted*. Restricted variables are defined as a particular subset of the variables close to their bounds, based on the sign of the corresponding components in the gradient. Formally, the set of *restricted* variables is

$$\mathfrak{I}^k \triangleq \big\{i \,\big|\, x_i^k \le l_i + \varepsilon \wedge \partial_i f(\boldsymbol{x}^k) > 0, \quad \text{or} \quad x_i^k \ge u_i - \varepsilon \wedge \partial_i f(\boldsymbol{x}^k) < 0\big\},$$

for some small positive ε. The set $\mathfrak{I}^k$ collects variables that are near their bounds, *and* for which the objective $f(\boldsymbol{x})$ can be decreased by moving the variables toward (or past) their bounds. The set of *free* variables, denoted $\mathfrak{F}^k$, is simply defined as the complement of $\mathfrak{I}^k$ in the set $\{1, 2, \dots, n\}$.

Without loss of generality, let us assume that $\mathfrak{F}^k = \{1, 2, \cdots, N\}$ and $\mathfrak{I}^k = \{N+1, \cdots, n\}$. Now define a diagonal matrix $\boldsymbol{D}^k \in \mathbb{R}^{n-N\times n-N}$ that scales the *restricted* variables, a typical choice being the identity matrix. We denote the scaling with respect to the free variables as $\bar{\boldsymbol{S}}^k \in \mathbb{R}^{N \times N}$, which, for the projected-Newton method, is given by the principal submatrix of the inverse of the Hessian $\nabla^2 f(\boldsymbol{x}^k)$, as induced by the free variables. In symbols, this is

$$\bar{\boldsymbol{S}}^k \leftarrow [\nabla^2 f(\boldsymbol{x}^k)]^{-1}_{\mathfrak{F}^k}. \tag{11.11}$$

With these definitions, we are now ready to present the main step of the two-metric projection algorithm. This step can be written as the Euclidean projection of a Newton step that uses a gradient scaling $\boldsymbol{S}^k$ of the form

$$\boldsymbol{S}^k \triangleq \begin{bmatrix} \bar{\boldsymbol{S}}^k & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}^k \end{bmatrix}. \tag{11.12}$$

The associated stepsize α can be selected by backtracking along the projection arc until the Armijo condition is satisfied. Note that this choice of the stepsize computation does not increase the computational complexity of the method, since computing the orthogonal projection after each backtracking step is trivial. Combining this gradient scaling with orthogonal projection, we obtain the projected Newton update:

$$\begin{aligned} \boldsymbol{x}^{k+1} \leftarrow \bar{\boldsymbol{x}}^k_\alpha &= \operatorname*{argmin}_{\boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}} \frac{1}{2} \|\boldsymbol{x} - (\boldsymbol{x}^k - \alpha \boldsymbol{S}^k \nabla f(\boldsymbol{x}^k))\|^2_{[\boldsymbol{S}^k]^{-1}} \\ &\approx \operatorname*{argmin}_{\boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}} \frac{1}{2} \|\boldsymbol{x} - (\boldsymbol{x}^k - \alpha \boldsymbol{S}^k \nabla f(\boldsymbol{x}^k))\|^2_{\boldsymbol{I}} \\ &= \mathcal{P}[\boldsymbol{x}^k - \alpha^k \boldsymbol{S}^k \nabla f(\boldsymbol{x}^k)], \end{aligned} \tag{11.13}$$

where α^k is computed by backtracking along the projection arc.

This algorithm has been shown to be globally convergent (Bertsekas, 1982; Gafni and Bertsekas, 1984), and under certain conditions achieves local superlinear convergence.

Theorem 11.1 (Convergence)**.** *Assume that ∇f is Lipschitz continuous on Ω, and $\nabla^2 f$ has bounded eigenvalues. Then every limit point of $\{\boldsymbol{x}^k\}$ generated by iteration* (11.13) *is a stationary point of* (11.10)*.*

Theorem 11.2 (Convergence rate)**.** *Let f be strictly convex and twice continuously differentiable. Let $\boldsymbol{x}^*$ be the non degenerate optimum of Problem* (11.13) *and assume that for some $\delta > 0$, $\nabla^2 f(\boldsymbol{x})$ has bounded eigenvalues for all $\boldsymbol{x}$ that satisfy $\|\boldsymbol{x} - \boldsymbol{x}^*\| < \delta$. Then the sequence $\{\boldsymbol{x}^k\}$ generated by iteration* (11.13) *converges to $\boldsymbol{x}^*$, and the rate of convergence in $\{\|\boldsymbol{x}^k - \boldsymbol{x}^*\|\}$ is superlinear.*

Although the convergence rate of the two-metric projection method has been shown for $\boldsymbol{S}^k$ derived from the Hessian, the convergence itself merely requires a positive definite gradient scaling $\boldsymbol{S}^k$ with bounded eigenvalues for all k (Bertsekas, 1982). Thus, the quasi-Newton approximations introduced in Section 11.2 are viable choices to derive convergent methods,[4] and we present such variations of the two-metric method in the following example.

Example 11.1 (Nonnegative least-squares)**.** *A problem of considerable importance in the applied sciences is the nonnegative least-squares (NNLS):*

$$\min_{\boldsymbol{x}} \quad \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2, \quad \textit{subject to } \boldsymbol{x} \geq 0, \tag{11.14}$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$. This problem is essentially an instance of (11.10)*.*

Given Algorithm 11.1, one can simply implement the update (11.13) *and then use* BFGS *or* L-BFGS *to obtain $\boldsymbol{S}^k$. However, we can further exploit the simple constraint $\boldsymbol{x} \geq \boldsymbol{0}$ and improve the computational (empirical) efficiency of the algorithm. To see how, consider the restricted variables in the update* (11.13)*. When variable $i \in \mathfrak{I}^k$ and ε becomes sufficiently small, we obtain*

$$\mathcal{P}[\boldsymbol{x}^k - \alpha^k \boldsymbol{S}^k \nabla f(\boldsymbol{x}^k)]_i = \mathcal{P}[x_i^k - \alpha^k [\boldsymbol{D}^k]_{ii} \cdot \partial_i f(\boldsymbol{x}^k)] = 0.$$

In other words, if $i \in \mathfrak{I}^k$, then $x_i^{k+1} = 0$, whereby we can safely ignore these variables throughout the update. In an implementation, this means that we can confine computations to free *variables, which can save a large number of floating point operations, especially when $|\mathfrak{F}^k| \ll |\mathfrak{I}^k|$.*

Example 11.2 (Linear SVM)**.** *Consider the standard binary classification task with inputs $(\boldsymbol{x}_i, y_i)_{i=1}^m$, where $\boldsymbol{x}_i \in \mathbb{R}^n$ and $y_i \in \pm 1$. Assume for simplicity that we wish to learn a bias-free decision function $f(\boldsymbol{x}) = sgn(\boldsymbol{w}^T\boldsymbol{x})$ by solving either the SVM* primal

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\textit{minimize}} & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^m \xi_i \\ \textit{subject to} & y_i(\boldsymbol{w}^T\boldsymbol{x}_i) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad 1 \leq i \leq m, \end{array} \tag{11.15}$$

or its (more familiar) dual

$$\begin{array}{ll} \underset{\boldsymbol{\alpha}}{\textit{minimize}} & \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{Y}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{Y}\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\boldsymbol{1} \\ \textit{subject to} & 0 \leq \alpha_i \leq C, \end{array} \tag{11.16}$$

4. In a simpler but still globally convergent variation of the two-metric projection method, we could simply set $\boldsymbol{S}^k$ to be a diagonal matrix with positive diagonal elements.

where $\boldsymbol{Y} = \mathrm{Diag}(y_1, \ldots, y_m)$ *and* $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m] \in \mathbb{R}^{n\times m}$. *The dual* (11.16) *is a special case of (11.10), and can be solved by adapting the two-metric projection method in a manner similar to that for NNLS.*

Example 11.3 (Sparse Gaussian graphical models)**.** *The dual to a standard formulation for learning sparse Gaussian graphical models takes the form (Banerjee et al., 2006)*

$$\min_{\tilde{\Sigma}+\boldsymbol{X}\succ 0} \quad -\log\det(\tilde{\Sigma}+\boldsymbol{X}), \quad \textit{subject to } |\boldsymbol{X}_{ij}| \leq \lambda_{ij}, \quad \forall_{ij}. \tag{11.17}$$

There has been substantial recent interest in solving this problem (Banerjee et al., 2006). Here $\tilde{\Sigma}$ *represents the empirical covariance of a data set, and the bound constraints on the elements of the matrix encourage the associated graphical model to be sparse for sufficiently large values of the* λ_{ij} *variables.*

Notice that the constraint $|\boldsymbol{X}_{ij}| \leq \lambda_{ij}$ *is equivalent to the bound constraints* $-\lambda_{ij} \leq \boldsymbol{X}_{ij} \leq \lambda_{ij}$. *Thus, provided* $\tilde{\Sigma}+\boldsymbol{X}$ *is positive definite for the initial* $\boldsymbol{X}$*, we can apply a simplified two-metric projection algorithm to this problem in which we use projection to address the bound constraints and backtracking to modify the iterates when they leave the positive definite cone.*

11.4 Inexact Projection Methods

The previous section focused on examples with bound constraints. For optimizing over more general but still simple convex sets, an attractive choice is *inexact* projected Newton methods. These methods represent a natural generalization of methods for unconstrained optimization that are alternatively referred to as *Hessian-free*, *truncated*, or *inexact* Newton methods. In inexact projected Newton methods, rather than finding the exact minimizer in Step IIa of Algorithm 11.1, we find an approximate minimizer using an iterative solver. That is, we use a single-metric projection, but solve the projection inexactly. Note that the iterative solver can be a first-order optimization strategy, and thus can take advantage of an efficient Euclidean projection operator. Under only mild conditions on the iterative solver, this approximate projection algorithm still leads to an improvement in the objective function. There are many ways to implement an inexact projected Newton strategy, but in this section we focus on the one described in Schmidt et al. (2009). In this method, we use the L-BFGS Hessian approximation, which we combine with simple Armijo backtracking and a variant of the projected gradient algorithm for iteratively solving subproblems. In Section 11.4.1 we review an effective iterative solver, and Section 11.4.2 we discuss using it within an inexact projected-Newton method.

11.4.1 Spectral Projected Gradient

The traditional motivation for examining projected Newton methods is that the basic gradient projection method may take a very large number of iterations to reach an acceptably accurate solution. However, there has been substantial recent interest in variants of gradient projection that exhibit much better empirical convergence properties. For example, Birgin et al. (2000) presented several *spectral* projected gradient (SPG) methods. In SPG methods, either α or β is set to 1, and the other stepsize is set to one of the stepsizes proposed by Barzilai and Borwein (1988). For example, we might set $\beta = 1$ and α to

$$\alpha^{bb} \triangleq \frac{\boldsymbol{g}^T\boldsymbol{s}}{\boldsymbol{g}^T\boldsymbol{g}}, \quad \text{where} \quad \boldsymbol{g} = \nabla f(\boldsymbol{x}^{k+1}) - \nabla f(\boldsymbol{x}^k), \text{ and } \boldsymbol{s} = \boldsymbol{x}^{k+1} - \boldsymbol{x}^k. \tag{11.18}$$

Subsequently, backtracking along one of the two stepsizes is used to satisfy the *non monotonic* Armijo condition (Grippo et al., 1986):

$$f(\boldsymbol{x}^{k+1}) \leq \max_{i=k-m:k} \{f(\boldsymbol{x}^i)\} + \tau \nabla f(\boldsymbol{x}^k)(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k), \qquad \tau \in (0,1).$$

Unlike the ordinary Armijo condition (11.5), allows some temporary increase in the objective. This non monotonic Armijo condition typically accepts the initial step length, even if it increases the objective function, while still ensuring global convergence of the method.[5] Experimentally, these two simple modifications lead to large improvements in the convergence speed of the method. Indeed, due to its strong empirical performance, SPG has recently been explored in several other applications (Dai and Fletcher, 2005; Figueiredo et al., 2007; van den Berg and Friedlander, 2008).

An alternative to SPG for accelerating the basic projected gradient method is the method of Nesterov (2004, Section 2.2.4). In this strategy, an extra extrapolation step is added to the iteration, thereby allowing the method to achieve the optimal worst-case convergence rate among a certain class of algorithms. Besides SPG and this optimal gradient algorithm, there can be numerous alternative iterative solvers. But we restrict our discussion to an SPG-based method and consider some implementation and theoretical details for it.

5. A typical value for the number m of previous function values to consider is 10.

11.4.2 SPG-based Inexact Projected Newton

Recall Step IIa in the general framework of Algorithm 11.1:[6]

$$\bar{\boldsymbol{x}}_1^k = \operatorname*{argmin}_{\boldsymbol{x}\in\Omega} \mathcal{Q}^k(\boldsymbol{x}, 1), \tag{11.19}$$

where the quadratic model is

$$\mathcal{Q}^k(\boldsymbol{x}, 1) = f(\boldsymbol{x}^k) + (\boldsymbol{x} - \boldsymbol{x}^k)^T \nabla f(\boldsymbol{x}^k) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^k)^T \boldsymbol{H}^k (\boldsymbol{x} - \boldsymbol{x}^k).$$

For the remainder of this section, we denote $\mathcal{Q}^k(\boldsymbol{x}, 1)$ by $\mathcal{Q}^k$ when there is no confusion. Inexact projected Newton methods solve the subproblem (11.19) only approximately; we denote this approximate solution by $\boldsymbol{z}^k$ below.

At each iteration of an SPG-based inexact projected Newton method, we first compute the gradient $\nabla f(\boldsymbol{x}^k)$ and (implicitly) compute the quadratic term $\boldsymbol{H}^k$ in $\mathcal{Q}^k$. Subsequently, we try to minimize this $\mathcal{Q}^k$ over the feasible set, using iterations of an SPG algorithm. Even if f or ∇f is difficult to compute, this SPG subroutine can be efficient if $\mathcal{Q}^k$ and $\nabla\mathcal{Q}^k$ can be evaluated rapidly. Given $f(\boldsymbol{x}^k)$ and $\nabla f(\boldsymbol{x}^k)$, the dominant cost in evaluating $\mathcal{Q}^k$ and $\nabla\mathcal{Q}^k$ is pre-multiplication by $\boldsymbol{H}^k$. By taking the compact representation of Byrd et al. (1994),

$$\boldsymbol{H}^k = \sigma^k \mathrm{I} - \boldsymbol{N}\boldsymbol{M}^{-1}\boldsymbol{N}^T, \quad \text{where} \quad \boldsymbol{N} \in \mathbb{R}^{n\times 2m},\ \boldsymbol{M} \in \mathbb{R}^{2m\times 2m}, \tag{11.20}$$

we can compute $\mathcal{Q}^k$ and $\nabla\mathcal{Q}^k$ in $\mathcal{O}(mn)$ under the L-BFGS Hessian approximation.

In addition to $\mathcal{Q}^k$ and $\nabla\mathcal{Q}^k$, the SPG subroutine also requires computing the Euclidean projection $\mathcal{P}_\Omega$ onto the feasible set Ω. However, note that the SPG subroutine does not evaluate f or ∇f. Hence, the SPG-based inexact projected Newton method is most effective on problems where computing the projection is much less expensive than evaluating the objective function.[7]

Although in principle we could use SPG to solve the problem (11.19) exactly, in practice this is expensive and ultimately unnecessary. Thus, we terminate the SPG subroutine before the exact solution is found. One might be concerned about terminating the SPG subroutine early, especially because an approximate solution to (11.3) will in general not be a descent

6. We assume that $\alpha = 1$ and backtrack along β so that the iterative solver is invoked only once for each iteration; however, the inexact Newton method does not rule out the possibility of fixing β (and invoking the iterative solver for each backtracking step).
7. This is different from many classical optimization problems such as quadratic programming, where evaluating the objective function may be relatively inexpensive but computing the projection may be as difficult as solving the original problem.

direction. Fortunately, we can guarantee that the SPG subroutine yields a descent direction even under early termination if we initialize it with $\boldsymbol{x}^k$ and we perform at least one SPG iteration. To see this, note that positive definiteness of $\boldsymbol{H}^k$ implies that a sufficient condition for $\boldsymbol{z}^k - \boldsymbol{x}^k$ to be a descent direction for some vector $\boldsymbol{z}^k$ is that $\mathcal{Q}^k(\boldsymbol{z}^k, \alpha^k) < f(\boldsymbol{x}^k)$, since this implies the inequality

$$(\boldsymbol{z}^k - \boldsymbol{x}^k)^T \nabla f(\boldsymbol{x}^k) < 0.$$

By substituting $\mathcal{Q}^k(\boldsymbol{x}^k, \alpha^k) = f(\boldsymbol{x}^k)$, we see that

$$\mathcal{Q}^k(\boldsymbol{z}^k, \alpha^k) < \mathcal{Q}^k(\boldsymbol{x}^k, \alpha^k) = f(\boldsymbol{x}^k),$$

where $\boldsymbol{z}^k$ is the first point satisfying the Armijo condition when we initialize SPG with $\boldsymbol{x}^k$.[8] In other words, if we initialize the SPG subroutine with $\boldsymbol{x}^k$, then the SPG iterate gives a descent direction after the first iteration, and every subsequent iteration. Thus, it can be safely terminated early. In an implementation we can parameterize the maximum number of the SPG iterations by c, which results in an $\mathcal{O}(mnc)$ iteration cost for the inexact Newton method, assuming that projection requires $\mathcal{O}(n)$ time.

Example 11.4 (Blockwise-sparse Gaussian graphical models)**.** *Consider a generalization of Example 11.3 where instead of constraining the absolute values of matrix elements, we constrain the norms of a disjoint set of groups (indexed by g) of elements:*

$$\min_{\tilde{\Sigma}+\boldsymbol{X}\succ 0} \quad \tfrac{1}{2}\log\det(\tilde{\Sigma}+\boldsymbol{X}), \quad \textit{subject to } \|\boldsymbol{X}_g\|_2 \leq \lambda_g, \forall_g. \tag{11.21}$$

This generalization is similar to those examined in Duchi et al. (2008) and Schmidt et al. (2009), and it encourages the Gaussian graphical model to be sparse across groups of variables (i.e., all edges in a group g will be either included in or excluded from the graph). Thus, formulation (11.21) *encourages the precision matrix to have a blockwise sparsity pattern. Unfortunately, this generalization can no longer be written as a problem with bound constraints, nor can we characterize the feasible set with a finite number of linear constraints (though it is possible to write the feasible set using quadratic constraints). Nevertheless, it is easy to compute the projection onto the norm constraints; to project a matrix $\boldsymbol{X}$ onto the feasible set with respect to the norm constraints we simply set $\boldsymbol{X}_g = \lambda_g / \|\boldsymbol{X}_g\|_2$ for each group g. Considering the potentially high cost of evaluating the log-determinant*

8. We will be able to satisfy the Armijo condition, provided that $\boldsymbol{x}^k$ is not already a minimizer.

function (and its derivative), this simple projection suggests that inexact projected Newton methods are well suited for solving (11.21).

11.5 Toward Nonsmooth Objectives

In this section we reconsider Problem (11.1), but unlike previous sections, we now allow $r(\boldsymbol{x}) \neq 0$. The resulting *composite optimization* problem occurs frequently in machine learning and statistics, especially with $r(\boldsymbol{x})$ being a *sparsity-promoting* regularizer (see e.g., Chapter 2).

How should we deal with the nondifferentiability of $r(\boldsymbol{x})$ in the context of Newton-like methods? While there are many possible answers to this question, we outline two simple but effective solutions that align well with the framework laid out so far.

11.5.1 Two-Metric Subgradient Projection Methods

We first consider the following special case of (11.1):

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \quad \mathcal{F}(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum\nolimits_i r_i(x_i), \tag{11.22}$$

where $r(\boldsymbol{x})$ has the separable form $r(\boldsymbol{x}) = \sum_i r_i(x_i)$ and each $r_i : \mathbb{R} \to \mathbb{R}$ is continuous and convex but not necessarily differentiable. A widely used instance of this problem is when we have $r_i(x_i) = \lambda_i |x_i|$ for fixed $\lambda_i > 0$, corresponding to ℓ_1-regularization. Note that this problem has a structure similar to the bound-constrained optimization problem (11.10); the latter has *separable constraints*, while problem (11.22) has a *separable nonsmooth* term. We can use separability of the nonsmooth term to derive a *two-metric subgradient projection* method for (11.22), that is analogous to the two-metric gradient projection method discussed in Section 11.3. The main idea is to choose an appropriately defined *steepest descent* direction and then to take a step resembling a two-metric projection iteration in this direction.

To define an appropriate steepest descent direction, we note that even though the objective in (11.22) is not differentiable, its directional derivatives always exist. Thus, analogous to the differentiable case, we can define the steepest descent direction as the direction that minimizes the directional derivative; among all vectors with unit norm, the steepest descent direction locally decreases the objective most quickly. This direction is closely related to the element of the subdifferential of a function $\mathcal{F}(\boldsymbol{x})$ with minimum norm.

Definition 11.1 (Mininum-norm subgradient)**.** *Let*

$$\boldsymbol{z}^k = \underset{\boldsymbol{z}\in\partial\mathcal{F}(\boldsymbol{x})}{\operatorname{argmin}} ||\boldsymbol{z}||_2. \tag{11.23}$$

Following an argument outlined in (Bertsekas et al., 2003, Section 8.4),[9] the steepest descent direction for a convex function $\mathcal{F}(\boldsymbol{x})$ at a point $\boldsymbol{x}^k$ is $-\boldsymbol{z}^k$, where the subdifferential of (11.22) is given by

$$\partial\mathcal{F}(\boldsymbol{x}) = \partial\{f(\boldsymbol{x}^k) + r(\boldsymbol{x}^k)\} = \nabla f(\boldsymbol{x}^k) + \partial r(\boldsymbol{x}^k).$$

Using the separability of $r(\boldsymbol{x})$, we see that the minimum-norm subgradient (11.23) with respect to a variable x_i, is given by

$$z_i^k = \begin{cases} 0, \quad \text{if} \quad -\nabla_i f(\boldsymbol{x}^k) \in \left\{\partial^- r_i(x_i^k),\ \partial^+ r_i(x_i^k)\right\}, \\ \min\left\{\left|\nabla_i f(\boldsymbol{x}^k) + \partial^- r_i(x_i^k)\right|,\ \left|\nabla_i f(\boldsymbol{x}^k) + \partial^+ r_i(x_i^k)\right|\right\}, \quad \text{otherwise,} \end{cases}$$

where the directional derivative $\partial^+ r_i(x_i^k)$ is given by

$$\partial^+ r_i(x_i^k) = \lim_{\delta\to 0^+} \frac{r_i(x_i^k + \delta) - r_i(x_i^k)}{\delta}.$$

The directional derivative $\partial^- r_i(x_i^k)$ is defined similarly, with δ going to zero from below. Thus, it is easy to compute $\boldsymbol{z}^k$ given $\nabla f(\boldsymbol{x}^k)$, as well as the left and right partial derivatives ($\partial^- r_i(x_i^k)$ and $\partial^+ r_i(x_i^k)$) for each $r_i(x_i^k)$. Observe that when $r_i(x_i^k)$ is differentiable, $\partial^- r_i(x_i^k) = \partial^+ r_i(x_i^k)$, whereby the minimum norm subgradient is simply $\nabla_i f(\boldsymbol{x}^k) + \nabla_i r(\boldsymbol{x}^k)$. Further, note that $\boldsymbol{z}^k = \boldsymbol{0}$ at a global optimum; otherwise $-\boldsymbol{z}^k$ yields a descent direction and we can use it in place of the negative gradient within a line search method.

Similar to steepest descent for smooth functions, a generalized steepest descent for nonsmooth functions may converge slowly, and thus we seek a Newton-like variant. A natural question is whether we can merely use a scaling matrix $\boldsymbol{S}^k$ to scale the steepest descent direction. Similar to the two-metric projection algorithm, the answer is "no" for essentially the same reason: in general a scaled version of the steepest descent direction may turn out to be an *ascent* direction.

However, we can still use a similar solution to the problem. If we make the positive definite scaling matrix $\boldsymbol{S}^k$ *diagonal* with respect to the variables x_i that are close to locations where $r_i(x_i)$ is nondifferentiable, then we can still ensure that the method generates descent directions. Thus, we obtain

9. Replacing maximization with minimization and concavity with convexity.

a simple Newton-like method for nonsmooth optimization that uses iterates of the form

$$\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k - \alpha \boldsymbol{S}^k \boldsymbol{z}^k. \tag{11.24}$$

Here, matrix $\boldsymbol{S}^k$ has the same structure as (11.12), but now the variables that receive a diagonal scaling are variables close to nondifferentiable values. Formally, the set of *restricted* variables is:

$$\mathfrak{I}^k \triangleq \bigl\{ i \bigm| \min_{d_i \in \mathcal{D}_i} |d_i - x_i| \leq \varepsilon \bigr\}, \tag{11.25}$$

where $\mathcal{D}_i$ is the (countable) set containing all locations where $r_i(x_i)$ is nondifferentiable.

In many applications where we seek to solve a problem of the form (11.22), we expect the function to be nondifferentiable with respect to several of the variables at a solution. Further, it may be desirable that intermediate iterations of the algorithm lie at nondifferentiable points. For example, these might represent sparse solutions if a nondifferentiability occurs at zero. In these cases, we can add a projection step to the iteration that encourages intermediate iterates to lie at points of nondifferentiability. Specifically, if a variable x_i crosses a point of nondifferentiability, we project onto the point of nondifferentiability. Since we use a diagonal scaling with respect to the variables that are close to points of nondifferentiability, this projection reduces to computing the Euclidean projection onto bound constraints, where the upper and lower bounds are given by the nearest upper and lower points of nondifferentiability. Thus, each iteration is effectively a two-metric subgradient projection iteration. To make our description concrete, let us look at a specific example below.

Example 11.5 (ℓ_1-Regularization). *A prototypical composite minimization problem in machine learning is the ℓ_1-regularized task*

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \quad f(\boldsymbol{x}) + \sum\nolimits_{i=1}^{n} \lambda_i |x_i|. \tag{11.26}$$

The scalars $\lambda_i \geq 0$ control the degree of regularization, and for sufficiently large λ_i, the parameter x_i is encouraged to be exactly zero.

To apply our framework, we need to efficiently compute the minimum norm subgradient $\boldsymbol{z}^k$ for (11.26)*; this gradient may be computed as*

$$z_i^k \triangleq \begin{cases} \nabla_i f(\boldsymbol{x}) + \lambda_i \operatorname{sgn}(x_i), & |x_i| > 0 \\ \nabla_i f(\boldsymbol{x}) + \lambda_i, & x_i = 0, \nabla_i f(\boldsymbol{x}) < -\lambda_i \\ \nabla_i f(\boldsymbol{x}) - \lambda_i, & x_i = 0, \nabla_i f(\boldsymbol{x}) > \lambda_i \\ 0, & x_i = 0, -\lambda_i \leq \nabla_i f(\boldsymbol{x}) \leq \lambda_i. \end{cases} \tag{11.27}$$

For this problem, the restrictied variable set (11.25) *corresponds to those variables sufficiently close to zero,* $\{i||x_i| \leq \varepsilon\}$. *Making* $\boldsymbol{S}^k$ *partially diagonal with respect to the restricted variables as before, we define the two-metric projection step for* ℓ_1*-regularized optimization as*

$$\boldsymbol{x}^{k+1} = \mathcal{P}_{\mathcal{O}}[\boldsymbol{x}^k - \alpha \boldsymbol{S}^k \boldsymbol{z}^k, \boldsymbol{x}^k]. \tag{11.28}$$

Here, the orthant projection (that sets variables to exactly zero) is defined as

$$\mathcal{P}_{\mathcal{O}}(\boldsymbol{y}, \boldsymbol{x})_i \triangleq \begin{cases} 0, & \textit{if } x_i y_i < 0, \\ y_i, & \textit{otherwise.} \end{cases}$$

Applying this projection is effective at sparsifying the parameter vector since it sets variables that change sign to exactly zero, and it also ensures that the line search does not cross points of nondifferentiability. Provided that $\boldsymbol{x}^k$ *is not stationary, the steps in* (11.28) *are guaranteed to improve the objective for sufficiently small* α. *The stepsize* α *is selected by a backtracking line search along the projection arc to satisfy a variant of the Armijo condition where the gradient is replaced by the minimum norm subgradient. If at some iteration the algorithm identifies the correct set of nonzero variables and then maintains the orthant of the optimal solution, then the algorithm essentially reduces to an unconstrained Newton-like method applied to the nonzero variables.*

In the two-metric projection algorithm for bound-constrained optimization the choice of the diagonal scaling matrix $\boldsymbol{D}^k$ *simply controls the rate at which very small variables move toward zero, and does not have a significant impact on the performance of the algorithm. However, the choice of* $\boldsymbol{D}^k$ *in the algorithm for* ℓ_1*-regularization can have a significant effect on the performance of the method, since if* $\boldsymbol{D}^k$ *is too large, we may need to perform several backtracking steps before the step length is accepted, while too small a value will require many iterations to set very small variables to exactly zero. One possibility is to compute the Barzilai-Borwein scaling* α^{bb} *of the variables given by* (11.18), *and set* $\boldsymbol{D}^k$ *to* $\alpha^{bb}\boldsymbol{I}$.

11.5.2 Proximal Newton-like Methods

The method of Section 11.5.1 crucially relies on separability of the nonsmooth function $r(\boldsymbol{x})$. For more general nonsmooth $r(\boldsymbol{x})$, an attractive choice is to tackle the nondifferentiability of $r(\boldsymbol{x})$ via proximity operators (Moreau, 1962; Combettes and Wajs, 2005; Combettes and Pesquet, 2009). These operators are central to forward-backward splitting methods (Combettes and Pesquet, 2009),[10] as well as to methods based on surrogate optimization (Figueiredo and Nowak, 2003; Daubechies et al., 2004), separable approximation (Wright et al., 2009), gradient-mapping (Nesterov, 2007), or to a proximal trust-region framework (Kim et al., 2010).

The idea of a proximity operator is simple. Let $r : X \subseteq \mathbb{R}^d \to (-\infty, \infty]$ be a lower semicontinuous, proper convex function. For a point $\boldsymbol{y} \in X$, the *proximity operator* for r applied to $\boldsymbol{y}$ is defined as

$$\operatorname{prox}_r(\boldsymbol{y}) = \underset{\boldsymbol{x} \in X}{\operatorname{argmin}} \quad \tfrac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + r(\boldsymbol{x}). \tag{11.29}$$

This operator generalizes the projection operator, since when $r(\boldsymbol{x})$ is the indicator function for a convex set C, (11.29) reduces to projection onto C. This observation suggests that we might be able to replace projection operators with proximity operators. Indeed, this replacement is done in *forward-backward splitting* methods, where one iterates

$$\boldsymbol{x}^{k+1} = \operatorname{prox}_{\alpha^k r}(\boldsymbol{x}^k - \alpha^k \nabla f(\boldsymbol{x}^k));$$

the iteration "splits" the update into differentiable (forward) and nondifferentiable (proximal or backward) steps. This method generalizes first-order projected gradient methods, and under appropriate assumptions it can be shown that the sequence $\{f(\boldsymbol{x}^k) + r(\boldsymbol{x}^k)\}$ converges to $f(\boldsymbol{x}^*) + r(\boldsymbol{x}^*)$, where x^* is a stationary point.

At this point, the reader may already suspect how we might use proximity operators in our Newton-like methods. The key idea is simple: build a quadratic model, but *only* for the differentiable part, and tackle the nondifferentiable part via a suitable proximity operator. This simple idea was also previously exploited by Wright et al. (2009) and Kim et al. (2010). Formally, we consider the regularized quadratic model

$$\mathcal{Q}^k(\boldsymbol{x}, \alpha) \triangleq f(\boldsymbol{x}^k) + (\boldsymbol{x} - \boldsymbol{x}^k)^T \nabla f(\boldsymbol{x}^k) + \frac{1}{2\alpha}(\boldsymbol{x} - \boldsymbol{x}^k)^T \boldsymbol{H}^k (\boldsymbol{x} - \boldsymbol{x}^k) + r(\boldsymbol{x}), \tag{11.30}$$

whose minimizer can be recast as the *generalized proximity operator*:

$$\operatorname{prox}_{\alpha \cdot r}^{\boldsymbol{H}^k}(\boldsymbol{y}^k) = \underset{\boldsymbol{x} \in \mathbb{R}^n}{\operatorname{argmin}} \quad \tfrac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}^k\|_{\boldsymbol{H}^k}^2 + \alpha r(\boldsymbol{x}), \tag{11.31}$$

where $\boldsymbol{y}^k = \boldsymbol{x}^k - \alpha[\boldsymbol{H}^k]^{-1}\nabla f(\boldsymbol{x}^k)$; observe that under the transformation

10. Including *iterative soft-thresholding* as a special case.

$\boldsymbol{x} \to [\boldsymbol{H}^k]^{1/2}\boldsymbol{x}$, (11.31) may be viewed as a standard proximity operator.

Using this generalized proximity operator, our Newton-like algorithm becomes

$$\boldsymbol{x}^{k+1} = \operatorname{prox}_{\alpha \cdot r}^{\boldsymbol{H}^k}(\boldsymbol{x}^k - \alpha[\boldsymbol{H}^k]^{-1}\nabla f(\boldsymbol{x}^k)). \tag{11.32}$$

If instead of the true inverse Hessian, we use $\boldsymbol{H}^k = \boldsymbol{I}$, iteration (11.32) degenerates to the traditional forward-backward splitting algorithm. Furthermore, it is equally straightforward to implement a quasi-Newton variant of (11.32), where, for example, $\boldsymbol{H}^k$ is obtained by an L-BFGS approximation to $\nabla^2 f(\boldsymbol{x}^k)$. Another practical choice might be an inexact quasi-Newton variant, where we use iterations of the SPG-like method of Wright et al. (2009) to approximately minimize $\mathcal{Q}^k(\boldsymbol{x}, \alpha)$ under an L-BFGS approximation of $f(\boldsymbol{x})$; in other words, the generalized proximity operator (11.31) is computed inexactly.

Similar to inexact projected Newton methods, under only mild assumptions we can guarantee that an inexact solution to the generalized proximity operator yields an improvement on the original objective for a sufficiently small stepsize α. For example, assume that $\nabla f(\boldsymbol{x})$ is Lipschitz continuous and that we find a value $\boldsymbol{y}$ such that $\mathcal{Q}^k(\boldsymbol{y}, \alpha) < \mathcal{Q}^k(\boldsymbol{x}^k, \alpha)$ in (11.30). Then we have

$$\begin{aligned} f(\boldsymbol{x}^k) + r(\boldsymbol{x}^k) &= \mathcal{Q}^k(\boldsymbol{x}^k, \alpha) \\ &> \mathcal{Q}^k(\boldsymbol{y}, \alpha) \\ &= f(\boldsymbol{x}^k) + (\boldsymbol{y} - \boldsymbol{x}^k)^T \nabla f(\boldsymbol{x}^k) + \frac{1}{2\alpha}(\boldsymbol{y} - \boldsymbol{x}^k)^T \boldsymbol{H}^k (\boldsymbol{y} - \boldsymbol{x}^k) + r(\boldsymbol{y}) \\ &\geq f(\boldsymbol{x}^k) + (\boldsymbol{y} - \boldsymbol{x}^k)^T \nabla f(\boldsymbol{x}^k) + \frac{m}{2\alpha}||\boldsymbol{y} - \boldsymbol{x}^k||_2^2 + r(\boldsymbol{y}) \\ &\geq f(\boldsymbol{y}) + r(\boldsymbol{y}) \quad (\text{for } 0 < \alpha \leq m/\mathcal{L}), \end{aligned}$$

where m is the smallest eigenvalue of $\boldsymbol{H}^k$ and the last inequality follows from Lipschitz continuity of the gradient (Bertsekas, 1999, Proposition A.24), where $\mathcal{L}$ is the Lipschitz constant of the gradient of $f(\boldsymbol{x})$. A similar property holds if $\nabla f(\boldsymbol{x})$ is only locally Lipschitz continuous.

Example 11.6 (Group ℓ_1-regularization)**.** *Consider a generalization of Example 11.5 where instead of penalizing the absolute values of each element of $\boldsymbol{x}$, we penalize the ℓ_2 norms of a set of disjoint groups indexed by g:*

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \quad f(\boldsymbol{x}) + \sum\nolimits_g \lambda_g \|\boldsymbol{x}_g\|_2. \tag{11.33}$$

The regularizer in (11.33) *is referred to as a* group regularizer *($\ell_{1,2}$-regularizer), since it encourages sparsity in terms of groups of variables,* *and dates back to Bakin (1999). The regularization term is nonsmooth when an entire group of variables is set to* $\mathbf{0}$. *However, the proximal operator for this regularizer is easily computed: given a vector* $\boldsymbol{y}$, *with groups* $\boldsymbol{y}_g$, *we simply set* $\boldsymbol{x}_g = (\boldsymbol{y}_g/\|\boldsymbol{y}_g\|_2)\max\{0,\ \|\boldsymbol{y}_g\|_2 - \alpha\lambda_g\}$. *Thus, inexact proximal Newton methods are well suited to solving* (11.33).

Example 11.7 (Group nuclear norm regularization)**.** *A related problem is optimizing a smooth function of several matrix inputs with regularization of the nuclear norms of the matrices:*

$$\min_{\boldsymbol{X}_1,\boldsymbol{X}_2,\ldots,\boldsymbol{X}_n} \quad f(\boldsymbol{X}_1,\boldsymbol{X}_2,\ldots,\boldsymbol{X}_n) + \sum\nolimits_{i=1}^{n} \lambda_i \|\boldsymbol{X}_i\|_*. \tag{11.34}$$

Here we use $\|\boldsymbol{X}\|_*$ *to the denote the* nuclear *norm (or* trace *norm), the sum of the singular values of* $\boldsymbol{X}$. *This regularization not only encourages sparsity across individual matrices, but also encourages each matrix to be low rank. The proximal operator for the nuclear norm can be computed by soft-thresholding the singular values of each* $\boldsymbol{X}_i$ *(Cai et al., 2010). That is, to compute the proximal operator, we replace each singular value* σ_j *of each* $\boldsymbol{X}_i$ *with* $\sigma_j = \max\{0,\ \sigma_j - \alpha\lambda_i\}$, *where* α *is the parameter of the quadratic approximation* (11.30). *Thus, inexact proximal Newton methods are well suited to solving* (11.34) *too, especially when it is more expensive to evaluate* f *and* ∇f *than it is to compute the singular value decomposition of each* $\boldsymbol{X}_i$.

11.6 Summary and Discussion

In this chapter, we have concentrated on minimizing twice-differentiable convex functions, both when their exact Hessian is feasible to use and when quasi-Newton choices are more practical. Note that the quasi-Newton approach can also be applied when the objective function is only once differentiable. Furthermore, we may relax the assumption of convexity if we concede that the stationary point found by the method may not be a local or global minimum.

As for implementational strategies, while we have focused on L-BFGS methods, an alternative restricted memory strategy is to use implicit Hessian-vector products. For example, in the two-metric projection strategy we can use Hessian-vector products within a linear conjugate gradient iteration to solve the scaling with respect to the free variables, as in Nocedal and Wright (2000, Section 7.1), while we can use Hessian-vector products within the SPG subroutine for inexact projected Newton methods. An alternative means to optimize nonsmooth objectives with an L-BFGS approximation is given by Yu et al. (2010). Variants of the L-BFGS approximation that apply in stochastic scenarios are examined in Sunehag et al. (2009).

We close by noting some open issues regarding convergence of the methods discussed in this section. First, global convergence of methods based on the minimum norm subgradient without a diminishing stepsize can be tenuous because of the lack of continuity in the derivatives of sequences. For example, see the counterexample in Bertsekas (1999, Exercise 6.3.8). Andrew and Gao (2007) give a proof of global convergence of a method related to the two-metric subgradient projection method we discuss in Section 11.5.1, but as pointed out by Yu et al. (2010), their proof does not account for this lack of continuity. Thus, while the algorithms of Andrew and Gao (2007) and Section 11.5.1 appear to be very effective in practice, it remains to be shown whether they are globally convergent in general without additional assumptions.

A related issue is showing whether the two-metric subgradient projection method identifies the correct set of nonzero variables after a finite number of iterations, and then has a superlinear convergence rate when exact second-order information is available. Also, in Examples 1.5 and 1.6, we use a projection with respect to a subset of the constraints and do not project with respect to the positive definite constraint that is known not to be active at the solution. Although this strategy has been used by several authors, and seems not to significantly affect empirical convergence of the method when given a suitable starting point, formally examining convergence under this heuristic deserves some theoretical attention.

Finally, there are not yet formal proofs of global and local convergence for inexact projected Newton methods, but this appears to be a simpler task than showing convergence of the methods discussed in the previous paragraph. For example, it is likely that global convergence can be proved by showing that a suitable gradient-related condition (Bertsekas, 1999, Section 1.2) applies to the first iteration in the SPG subroutine that satisfies the Armijo condition, while a local convergence rate can likely be shown by using a forcing sequence (Nocedal and Wright, 2000, Section 7.1) on the solution accuracy of the SPG subroutine.

11.7 References

G. Andrew and J. Gao. Scalable training of L_1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40, 2007.

S. Bakin. *Adaptive regression and model selection in data mining problems.* PhD thesis, Australian National University, Canberra, 1999.

O. Banerjee, L. El Ghaoui, A. d'Aspremont, and G. Natsoulis. Convex optimization techniques for fitting sparse Gaussian graphical models. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 89–96, 2006.

J. Barzilai and J. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

D. P. Bertsekas. Projected Newton methods for optimization problems with simple constraints. *SIAM Jounal on Control and Optimization*, 20(2):221–246, 1982.

D. P. Bertsekas. *Nonlinear Programming.* Athena Scientific, second edition, 1999.

D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar. *Convex Analysis and Optimization.* Athena Scientific, 2003.

E. G. Birgin, J. M. Martínez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10(4):1196–1211, 2000.

R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(1):129–156, 1994.

J. F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

P. L. Combettes and J. Pesquet. Proximal Splitting Methods in Signal Processing. *arXiv:0912.3522v2*, December 2009.

P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling and Simulation*, 4(4):1168–1200, 2005.

Y. H. Dai and R. Fletcher. Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming. *Numerische Mathematik*, 100(1):21–47, 2005.

I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

J. Duchi, S. Gould, and D. Koller. Projected subgradient methods for learning sparse gaussians. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 145–152, 2008.

M. Figueiredo, R. Nowak, and S. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, 2007.

M. A. T. Figueiredo and R. D. Nowak. An EM algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12(8):906–916, 2003.

E. M. Gafni and D. P. Bertsekas. Two-metric projection methods for constrained optimization. *SIAM Journal on Control and Optimization*, 22(6):936–964, 1984.

P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, 1981.

L. Grippo, F. Lampariello, and S. Lucidi. A nonmonotone line search technique for Newton's method. *SIAM Journal on Numerical Analysis*, 23(4):707–716, 1986.

D. Kim, S. Sra, and I. S. Dhillon. A scalable trust-region algorithm with application to mixed-norm regression. In *Proceedings of the 27th International Conference on Machine Learning*, pages 519–526, 2010.

E. S. Levitin and B. T. Polyak. Constrained minimization methods. *USSR Computational Mathematics and Mathematical Physics*, 6:1–50, 1966. English translation of a paper in Zh. Vȳchisl. Mat. i Mat. Fiz. 6, 5, 787-823, 1966.

J.-J. Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertien. *C. R. Acad. Sci. Paris*, 255:2897–2899, 1962.

Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer, 2004.

Y. Nesterov. Gradient methods for minimizing composite objective function. Technical report, Université Catholique de Louvain, 2007.

J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, second edition, 2000.

M. Schmidt, E. van den Berg, M. Friedlander, and K. Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-Newton algorithm. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, volume 5, pages 456–463, 2009.

D. F. Shanno and K. H. Phua. Matrix conditioning and nonlinear optimization. *Mathematical Programming*, 14(1):149–160, 1978.

P. Sunehag, J. Trumpf, S. V. N. Vishwanathan, and N. N. Schraudolph. Variable metric stochastic approximation theory. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, volume 5, pages 560–566, 2009.

E. van den Berg and M. P. Friedlander. Probing the Pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008.

S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

J. Yu, S. V. N. Vishwanathan, S. Günter, and N. N. Schraudolph. A quasi-newton approach to nonsmooth convex optimization. *Journal of Machine Learning Research*, 11:1145–1200, 2010.