Tractable optimization in machine learning*

Suvrit Sra†
Max Planck Institute for Intelligent Systems, 72076 Tübingen, Germany.

April 28, 2012

1 Introduction

Machine Learning (ML) broadly encompasses a variety of adaptive, autonomous, and intelligent tasks where one must "learn" to predict from observations and feedback. Throughout its evolution, ML has drawn heavily and successfully on optimization algorithms. This relation to optimization is not surprising as "learning" and "adapting" usually lead to problems where some quality function must be optimized.

But the interaction between ML and optimization is now undergoing rapid change. The increased size, complexity, and variety seen in ML problems, not only prompt a refinement of existing optimization techniques but also spur development of new methods tuned to the specific needs of ML applications.[1]

In particular, ML applications must usually cope with large-scale data, which forces us to prefer "simpler," perhaps less accurate but more scalable algorithms. Such methods can also crunch through more data, and may actually be better suited for learning—for a more precise characterization see [Bottou and Bousquet, 2011]. The use of possibly less accurate methods is also grounded in pragmatic realities: modeling limitations, observational noise, uncertainty, and computational errors are pervasive in real data. Hence, trusting more than a few digits of numerical accuracy would be unrealistic. From an engineering perspective, simpler algorithms translate into more reliable software that is easier to implement, debug, and deploy.

Before we get carried away by these benefits, we must recall a sobering statement of Nesterov [2004]: *"in general, optimization problems are unsolvable."* In other words, obtaining globally optimal solutions is in general intractable. Fortunately, nature makes a generous exception for *convex optimization*, which is not only tractable [Nemirovsky and Yudin, 1983] but also widely applicable [Boyd and Vandenberghe, 2004].

We, therefore, limit our attention to convex optimization, and therein we focus on algorithms that are simple, scalable, and amenable to theoretical analysis that qualifies their tractability.

The superficiality of a summary such as the one attempted herein is ineluctable. Nevertheless, we hope that it still provides a quick entry into large-scale convex optimization for non-experts, while offering pointers to literature that even more experienced readers might find useful.

*A shorter version of this paper will appear as a chapter of the same name in *"Advances in Tractability"* edited by L. Bordeaux and Y. Hamadi and P. Kohli and R. Mateescu.

†Part of this work was done during the author's visit at the EE Department, University of Washington, Seattle.

[1]This viewpoint is not limited to ML; other problem domains that deal with large-scale data intensive problems (e.g., bioinformatics, astroinformatics, signal processing) face similar concerns.

2 Background

A generic convex optimization problem may be written as

$$\min \quad \phi(\boldsymbol{x}) \quad \text{subject to } \boldsymbol{x} \in \mathcal{X}, \tag{1}$$

where $\phi : \mathbb{R}^n \to \mathbb{R}$ is a proper convex function, also called the *cost* or *objective function*, and $\mathcal{X}$ is a nonempty, compact convex set, also called the *constraint* or *feasible set*. In this chapter, we talk less about casting ML problems into the form (1); we focus more on algorithms for solving important instances of (1).

Example 2.1 (NNLS)**.** Suppose we wish to estimate a nonnegative signal (such as, frequency, probability, intensity) from a noisy measurement; for simplicity, assume that the measurement process is linear. Let the measurement be given by vector $\boldsymbol{b} \in \mathbb{R}^m$, and the linear process be encoded by a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. We wish to estimate an underlying nonnegative signal $\boldsymbol{x}$ that satisfies $\boldsymbol{b} \approx \boldsymbol{A}\boldsymbol{x}$. Assuming the noise to be additive and Gaussian, and that $\boldsymbol{A}^T\boldsymbol{A}$ is invertible. Then, we may estimate $\boldsymbol{x}$ by solving the *nonnegative least-squares* (NNLS) problem:

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2, \quad \text{s.t.} \quad \boldsymbol{x} \geq 0, \tag{2}$$

which is essentially of the form (1), since the optimal $\boldsymbol{x}$ can be shown to lie in a compact set.

To avoid triviality, assume that (1) has a solution—that is, there exists a point $\boldsymbol{x}^* \in \mathcal{X}$ for which $\phi(\boldsymbol{x}^*) \leq \phi(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$ (compactly, $\phi^* \leq \phi$). Ideally, the goal of an optimization algorithm is to compute $\boldsymbol{x}^*$, but almost always it is impossible to compute an *exact* $\boldsymbol{x}^*$ in finite time. Thus, we speak of "ϵ-accuracy," a term whose meaning may vary. Consider for instance, an *unconstrained* version of (1) where $\mathcal{X} = \mathbb{R}^n$. For differentiable ϕ, the condition $\nabla\phi(\boldsymbol{x}^*) = 0$ is necessary and sufficient for $\boldsymbol{x}^*$ to be a global minimizer. So an ϵ-accurate solution could be a point $\bar{\boldsymbol{x}}$, for which $\|\nabla\phi(\bar{\boldsymbol{x}})\| \leq \epsilon$ (we use $\|\boldsymbol{x}\| := \sqrt{\sum_i x_i^2}$). Alternatively, we may seek a point $\bar{\boldsymbol{x}}$ for which $\|\bar{\boldsymbol{x}} - \boldsymbol{x}^*\| \leq \epsilon$, or more often, for which the difference $\phi(\bar{\boldsymbol{x}}) - \phi(\boldsymbol{x}^*) \leq \epsilon$.

Intuitively, the tighter the accuracy ϵ, the harder it will be to compute an ϵ-accurate solution. Can this notion be made precise? Are there algorithms that can actually compute such solutions? These questions lie at the heart of the complexity theory of convex optimization, as pioneered by Yudin and Nemirovskii [1976]; see also [Nemirovsky and Yudin, 1983, Nesterov and Nemirovski, 1994]. For a nice historical account see [Tikhomirov, 1996].

2.1 Information based complexity

Observe that since we are working over the reals, the usual Turing machine based complexity analysis does not apply. It is more convenient to perform *information-based* complexity analysis using an *oracle model* of computation. More specifically, we consider algorithms that optimize (1) iteratively; at each iteration the algorithm queries a *first-order oracle*[2] using its current guess $\boldsymbol{x}$; the oracle responds by outputting the pair $(\phi(\boldsymbol{x}), \phi'(\boldsymbol{x}))$—here ϕ' denotes the gradient $\nabla\phi$ for differentiable ϕ, or a subgradient from the subdifferential[3] $\partial\phi(\boldsymbol{x})$, otherwise. We also speak of a *noisy first-order oracle*, which does not output ϕ and ϕ' exactly but rather their unbiased estimates.

We measure complexity of an algorithm by the number of queries that it makes for computing an ϵ-accuracy solution. If this number is at most a polynomial (in $1/\epsilon$), then the problem is called

[2]We may also speak of zeroth, second-order, or other types of oracles, but for brevity we omit these from our discussion.

[3]Please refer to [Rockafellar, 1970] for this and other key ideas of convex analysis.

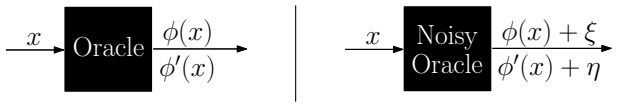

Figure 1: Noiseless and noisy first-order oracles. The noisy oracle returns values perturbed by mean-zero bounded variance noise. More generally, these oracles may receive / use any of their previous inputs or outputs when creating an output for the current $\boldsymbol{x}$.

tractable. Different tractable problems have different lower-complexity bounds on the number of queries that must be made; and an algorithm that achieves this lower bound (for all problems in its class) is called *optimal*.

This chapter describes some fundamental tractable and optimal algorithms; we divide our presentation into two parts: (i) smooth (at least once differentiable), and (ii) nonsmooth (but continuous) convex optimization. We discuss both noise-free and noisy oracles while highlighting both classical and modern results. Some concrete examples are interspersed to make our presentation clearer.

3 Smooth Convex Optimization

We begin with an instructive setup: unconstrained, smooth convex optimization for the class of *Lipschitz continuous* functions.

Definition 3.1 (Lipschitz continuity)**.** Let $\phi : \mathbb{R}^n \to \mathbb{R}$ be k times continuously differentiable on $\mathcal{X} \subseteq \mathbb{R}^n$. If the k-th derivative $\phi^{(k)}$ satisfies

$$\|\phi^{(k)}(\boldsymbol{x}) - \phi^{(k)}(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|, \quad \forall\ \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}, \tag{3}$$

for some constant L, then ϕ is said to have a *Lipschitz continuous* k-th derivative, and we denote this by writing $\phi \in C^k_L(\mathcal{X})$.

Example 3.2. Some familiar functions satisfying (3) are (verify!):

- *Linear:* $\langle \boldsymbol{a}, \boldsymbol{x} \rangle + c$ lies in $C^0_{\|\boldsymbol{a}\|}(\mathbb{R}^n)$ and also belongs to $C^1_0(\mathbb{R}^n)$
- *Hinge loss:* $\max(0, 1 - x)$ lies in $C^0_1(\mathbb{R})$;
- *Logistic loss:* $\log(1 + e^x)$ lies in $C^1_2(\mathbb{R})$;
- *Quadratic loss:* $\frac{1}{2}\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x}$ lies in $C^1_{\|A\|}(\mathbb{R}^n)$

Note that the *Log-loss*: $\phi(x) = -\log x \notin C^1_L(\mathbb{R}_{++})$ for any finite L.

Among Lipschitz continuous functions, the class $C^1_L(\mathcal{X})$ of functions with Lipschitz continuous gradients is of great importance. Most of the above examples lie in this class; and several differentiable (not necessarily convex) optimization problems encountered in machine learning and related areas feature C^1_L functions. One reason why this class enjoys such importance is the following very useful lemma.

Lemma 3.3 (Descent lemma)**.** *Let $\phi \in C^1_L(\mathcal{X})$. Then, it holds that*

$$|\phi(\boldsymbol{x}) - \phi(\boldsymbol{y}) - \langle \nabla\phi(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle| \leq \tfrac{1}{2}L\|\boldsymbol{x} - \boldsymbol{y}\|^2, \quad \forall\ \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}. \tag{4}$$

Proof. Since f is differentiable, we have the Taylor-expansion

$$\begin{aligned}\phi(\boldsymbol{x}) &= \phi(\boldsymbol{y}) + \textstyle\int_0^1 \langle \nabla\phi(\boldsymbol{y} + t(\boldsymbol{x}-\boldsymbol{y})),\, \boldsymbol{x}-\boldsymbol{y}\rangle dt \\ &= \phi(\boldsymbol{y}) + \langle \nabla\phi(\boldsymbol{y}),\, \boldsymbol{x}-\boldsymbol{y}\rangle + \textstyle\int_0^1 \langle \nabla\phi(\boldsymbol{y} + t(\boldsymbol{x}-\boldsymbol{y})) - \nabla\phi(\boldsymbol{y}),\, \boldsymbol{x}-\boldsymbol{y}\rangle dt.\end{aligned}$$

Rearranging, taking absolute values, invoking the triangle and Cauchy-Schwarz inequalities, along with Definition (3), we obtain

$$|\phi(\boldsymbol{x}) - \phi(\boldsymbol{y}) - \langle \nabla\phi(\boldsymbol{y}),\, \boldsymbol{x}-\boldsymbol{y}\rangle| \leq L\|\boldsymbol{x}-\boldsymbol{y}\|^2 \textstyle\int_0^1 t dt = \frac{1}{2}L\|\boldsymbol{x}-\boldsymbol{y}\|^2. \qquad \square$$

Now we look at optimization methods for convex $C_L^1(\mathcal{X})$ functions.

3.1 Gradient based methods

For solving (1), perhaps the simplest general purpose algorithm is *gradient-projection*; summarized by (5) below.

Initialize $\boldsymbol{x}_0 \in \mathbb{R}^n$
For $k \geq 0$ iterate:
Select suitable stepsize $\alpha_k > 0$
$\boldsymbol{x}_{k+1} = P_{\mathcal{X}}(\boldsymbol{x}_k - \alpha_k \nabla\phi(\boldsymbol{x}_k))$.

(5)

Iteration (5) consists of three key components: (i) the gradient $\nabla\phi$; (ii) the stepsize $\alpha_k > 0$; and (iii) the *projection operator*

$$P_{\mathcal{X}}(\boldsymbol{y}) := \operatorname*{argmin}_{\boldsymbol{x}\in\mathcal{X}} \tfrac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2. \tag{6}$$

As per our assumption, the gradient $\nabla\phi$ is generated by an oracle. The stepsize α_k can be set using various well-known strategies [Bertsekas, 1999, Nesterov, 2004], so we do not discuss it further. Finally, it is important to note that unless $\mathcal{X}$ is "simple", applying projection operator (6) can be as difficult as solving the overall problem itself.

Therefore, for simplicity we first analyze iteration (5) for the *unconstrained* case $\mathcal{X} = \mathbb{R}^n$, for which $P_{\mathcal{X}} \equiv \mathrm{Id}$. Here, we answer two key questions: (i) what is an *upper bound* on the number of oracle calls (iterations) needed by (5) to obtain an ϵ-accuracy solution; and (ii) how far is this upper bound from the *lower bound* on oracle calls?

Theorem 3.4 (Upper bound)**.** *Let ϕ be a convex, $C_L^1(\mathbb{R}^n)$ function. Let $\alpha_k = 1/L$ and let $\boldsymbol{x}^*$ be an optimal solution to* (1)*. Define the "diameter" $D := \|\boldsymbol{x}_0 - \boldsymbol{x}^*\|$. Then, the iterates $\{\boldsymbol{x}_k\}$ of* (5) *satisfy*

$$\phi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}^*) \leq \frac{2LD^2}{k+4}. \tag{7}$$

Proof. We follow [Nesterov, 2004, Theorem 2.1.14], and present the details for their instructive value. Define the residue $r_k = \|\boldsymbol{x}_k - \boldsymbol{x}^*\|$, let $\boldsymbol{g}_k \equiv \nabla\phi(\boldsymbol{x}_k)$, and $\boldsymbol{g}^* = \nabla\phi(\boldsymbol{x}^*)$. Recall now the optimality condition

$$\langle \boldsymbol{g}_k - \boldsymbol{g}^*,\, \boldsymbol{x}_k - \boldsymbol{x}^*\rangle \geq 0 \qquad \text{for all } \boldsymbol{x}_k.$$

This, together with the constant stepsize $\alpha_k = 1/L$, implies that

$$r_{k+1}^2 = \|\boldsymbol{x}_k - \boldsymbol{x}^* - \alpha_k \boldsymbol{g}_k\|^2 \quad \leq \quad r_k^2 - \tfrac{1}{L}\|\boldsymbol{g}_k\|^2,$$

and in particular that, $r_k \leq r_0 = D$. Now let $\Delta_k := \phi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}^*)$, and use convexity of ϕ and the Cauchy-Schwarz inequality to obtain

$$\Delta_k \leq \langle \boldsymbol{g}_k, \boldsymbol{x}_k - \boldsymbol{x}^* \rangle \leq \|\boldsymbol{g}_k\| \|\boldsymbol{x}_k - \boldsymbol{x}^*\| = \|\boldsymbol{g}_k\| r_k \leq \|\boldsymbol{g}_k\| D. \tag{8}$$

Invoke Lemma 3.3 with $\boldsymbol{x} = \boldsymbol{x}_{k+1}$ and $\boldsymbol{y} = \boldsymbol{x}_k$, to now see that

$$\phi(\boldsymbol{x}_{k+1}) \leq \phi(\boldsymbol{x}_k) + \langle \boldsymbol{g}_k, \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle + \tfrac{1}{2} L \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|^2.$$

Then use $\boldsymbol{x}_{k+1}$ from iteration (5) with $\alpha_k = 1/L$ to obtain

$$\phi(\boldsymbol{x}_{k+1}) \leq \phi(\boldsymbol{x}_k) - \tfrac{1}{2L} \|\boldsymbol{g}_k\|^2,$$

which implies the inequality $\Delta_{k+1} \leq \Delta_k - (1/2L)\|\boldsymbol{g}_k\|^2$. From inequality (8) and Lemma 3.3 we further obtain the bound

$$\Delta_{k+1} \leq \Delta_k - \Delta_k^2/(2LD^2) = \Delta_k(1 - \beta),$$

for some $\beta \in (0, 1)$. Take reciprocals and note that $(1 - \beta)^{-1} \leq 1 + \beta$, which yields $\frac{1}{\Delta_{k+1}} \geq \frac{1+\beta}{\Delta_k} = \frac{1}{\Delta_k} + \frac{1}{2LD^2}$. Summing this over k we obtain

$$\frac{1}{\Delta_{k+1}} \geq \frac{1}{\Delta_0} + \frac{k+1}{2LD^2},$$

which upon rearranging yields

$$\phi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}^*) \leq \frac{2LD^2\Delta_0}{2LD^2 + k\Delta_0} \leq \frac{2LD^2}{k+4}. \tag{9}$$

The final inequality follows because $\Delta_0 \leq \frac{1}{2}LD^2$ (Lemma 3.3) and because the second-last term in (9) is an increasing function of Δ_0. □

Corollary 3.5. *The gradient method* (5) *requires* $O(1/\epsilon)$ *iterations (calls to the oracle) to obtain a solution of* ϵ*-accuracy.*

Proof. From Theorem 3.4 it follows that if $k + 4 \geq \frac{2LD^2}{\epsilon}$, then $\phi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}^*) \leq \epsilon$; thus, $O(1/\epsilon)$ iterations yield a solution of ϵ-accuracy. □

Let us now state a lower-bound on the number of oracle calls. In particular, this bound applies to methods that generate the k-th iterate $\boldsymbol{x}_k$ by linearly combining the information obtained from the oracle up to step $k-1$. These methods generate iterates that satisfy

$$\boldsymbol{x}_k \in \boldsymbol{x}_0 + \mathrm{Lin}(\nabla\phi(\boldsymbol{x}_0), \ldots, \nabla\phi(\boldsymbol{x}_{k-1})). \tag{10}$$

Theorem 3.6 (Lower bound)**.** *Using the same notation as in Theorem 3.4, and assuming that* $1 \leq k \leq \frac{1}{2}(n-1)$*, there exists a convex function* $\phi \in C_L^1$*, such that any method that generates a sequence* $\{\boldsymbol{x}_k\}$ *satisfying* (10)*, must also satisfy*

$$\phi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}^*) \geq \frac{3LD^2}{32(k+1)^2}. \tag{11}$$

Proof. See [Nesterov, 2004, Section 2.1.2]. □

Both the upper and lower accuracy bounds show (slow) sublinear convergence. So a natural question is if there is a subclass of C^1_L functions for which we can obtain faster convergence rates? It turns out that for the subclass of *strongly convex* functions, we do have better rates.

Definition 3.7 (Strong convexity)**.** A function $\phi : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$ is said to be *strongly convex* with parameter $\mu > 0$, denoted $\phi \in S^1_{L,\mu}(\mathcal{X})$, if

$$\phi(\boldsymbol{x}) \geq \phi(\boldsymbol{y}) + \langle \nabla\phi(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y}\rangle + \tfrac{1}{2}\mu\|\boldsymbol{x} - \boldsymbol{y}\|^2, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}. \tag{12}$$

For such functions we have the following upper and lower bounds.

Theorem 3.8 (SC upper-bound)**.** *Let $\phi \in S^1_{L,\mu}(\mathbb{R}^n)$, and set $\alpha_k = 2/(L+\mu)$. Then, the iteration* (5) *generates a sequence $\{\boldsymbol{x}_k\}$ such that*

$$\phi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}^*) \leq \frac{L}{2}\Big(\frac{\kappa - 1}{\kappa + 1}\Big)^{2k} D^2, \tag{13}$$

where $\kappa = L/\mu$ denotes the "condition number" of ϕ.

Theorem 3.9 (SC lower-bound)**.** *There exists an infinitely differentiable $S^1_{\mu,\mu\kappa}$ function such that any first-order method that generates $\{\boldsymbol{x}_k\}$ satisfying (10) must also satisfy*

$$\phi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}^*) \geq \frac{\mu}{2}\Big(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\Big)^{2k} D^2. \tag{14}$$

For proofs of these theorems, refer to [Nesterov, 2004, Section 2.1.4].

Comparing the upper-bounds (7) and (13) with the lower-bounds (11) and (14), respectively, we see that gradient-descent is far from optimal. In a breakthrough paper Nesterov [1983] introduced a new *optimal gradient method* whose upper complexity bound matches (up to constants) the above lower bound. While the derivation of the optimal method and its analysis lie outside the scope of this chapter, some intuitive insight may be obtained by recalling the *heavy-ball method* [Polyak, 1987]. For a fixed $\gamma > 0$, this method performs the iteration

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla\phi(\boldsymbol{x}_k) + \gamma(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}). \tag{15}$$

Let ϕ be an $S^2_{L,\mu}$ function. Set $\alpha_k = \frac{4}{\mu}\frac{1}{(1+\sqrt{\kappa})^2}$ and $\gamma = (\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^2$; then, some algebra reveals that the "multistep" iteration (15) satisfies

$$\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2 \leq \Big(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\Big)^{2k} D^2.$$

Notice that this matches the optimal rate (14) (up to constants).

Thus, we might expect a multistep iteration to be effective also for $C^1_L(\mathbb{R}^n)$ functions. Nesterov's optimal method is such an iteration, though with a key difference: it uses iteration dependent values of γ. Here's how.

Let $\boldsymbol{y}_0 = \boldsymbol{x}_0$, $\alpha_0 \geq 1/\sqrt{\kappa}$
For $k \geq 0$ iterate:

$$\left\lfloor \begin{array}{ll} \boldsymbol{x}_{k+1} & = \boldsymbol{y}_k - \frac{1}{L}\nabla\phi(\boldsymbol{y}_k) \\ \alpha_{k+1} & = \frac{1}{2\kappa} - \frac{\alpha_k^2}{2} + \sqrt{\alpha_k^2 + \left(\frac{1}{2\kappa} - \frac{\alpha_k^2}{2}\right)^2} \end{array} \right. \tag{16a}$$

$$\left\lfloor \begin{array}{ll} \gamma_k & = \frac{\alpha_k(1-\alpha_k)}{\alpha_k^2 + \alpha_{k+1}} \\ \boldsymbol{y}_{k+1} & = \boldsymbol{x}_{k+1} + \gamma_k(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k). \end{array} \right. \tag{16b}$$

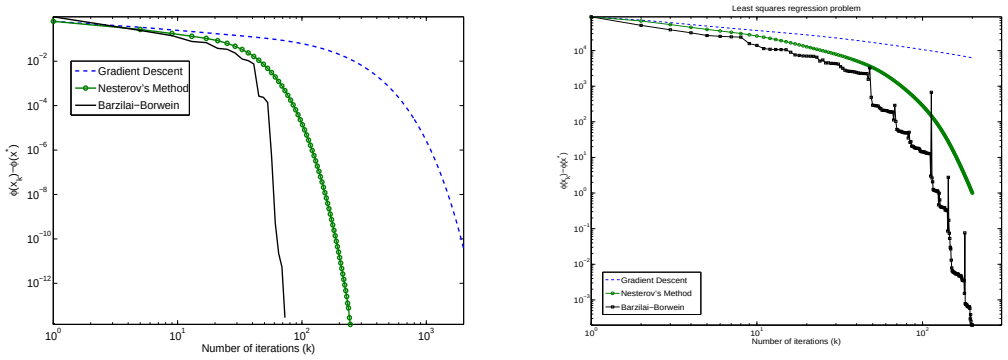

Figure 2: Gradient-descent (5), Nesterov's method (16), and gradient-descent with Barzilai-Borwein (BB) stepsizes (18). In the left plot we minimize $\phi(\boldsymbol{x}) = 0.5\|\boldsymbol{D}^T\boldsymbol{x} - \boldsymbol{b}\|^2$, with $\boldsymbol{D}$ being an $(n-1)\times n$ matrix having -1s on its diagonal and 1s on the first superdiagonal. We used $n = 50$. The vector $\boldsymbol{b}$ is set to $\mathbf{2}$ (all twos). Here, $L = \lambda_{\max}(\boldsymbol{D}\boldsymbol{D}^T) \approx 4$, and $\mu = \lambda_{\min}(\boldsymbol{D}\boldsymbol{D}^T) \approx 1/(0.1n^2 + 0.6n)$ (so $\boldsymbol{D}$ is very ill-conditioned). The stepsize $\alpha_k = 1/L$ is used for (5). The superiority of Nesterov's optimal method over gradient descent is evident; surprisingly, the BB method performs even better. The right plot runs a least squares problem similar to the left plot, but this time on the UCI Year-Prediction dataset [Frank and Asuncion, 2010]. Specifically, we solve $\min_{\boldsymbol{w}} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2$ on a $463,715 \times 90$ data matrix $\boldsymbol{X}$ (normalized to have unit columns); here $L/\mu \approx 933$.

The similarity of iteration (16b) to (15) is unmistakable. Theorem 3.10 characterizes the optimality of the above method.

Theorem 3.10. *Let $\{\boldsymbol{x}_k\}$ be generated by (16) with $\alpha_0 = \hat{\alpha} + \sqrt{1+\hat{\alpha}^2}$, where $\hat{\alpha} = -1/2 + 1/(2\kappa)$. Then, we have the upper bound:*

$$\phi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}^*) \leq \frac{3}{2}LD^2 \times \min\Big\{\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}}\Big)^k, \frac{4}{(k+2)^2}\Big\}. \tag{17}$$

Proof. Use $\gamma_0 = L$ in [Nesterov, 2004, Theorem 2.2.3] along with the inequality (follows from Lemma 3.3) that $\phi(\boldsymbol{x}_0) - \phi(\boldsymbol{x}^*) \leq \frac{1}{2}LD^2$. □

Comparing (17) with the lower bound (11) one sees that for general $C^1_L(\mathbb{R}^n)$ convex functions, method (16) is optimal to within constant factors. For the strongly convex case, the lower bound (14) implies that if $\phi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}^*) \leq \epsilon$, then $2k\log\Big(1 + \frac{2}{\sqrt{\kappa}-1}\Big) \geq \log\Big(\frac{\mu D^2}{2\epsilon}\Big)$, whereby

$$k \geq \tfrac{\sqrt{\kappa}-1}{4}\big(\log(\tfrac{1}{\epsilon} + \log\tfrac{\mu D^2}{2})\big)$$

iterations are needed to obtain an ϵ-accuracy solution. Since $\phi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}^*) \leq (3/2)LD^2(1-1/\sqrt{\kappa})^k$, using the inequality $-\log(1-1/\sqrt{\kappa}) < 1/\sqrt{\kappa}$ for $\kappa > 1$, we conclude that $k \leq \sqrt{\kappa}\big(\log\frac{1}{\epsilon} + \log\frac{3LD^2}{2}\big)$ iterations suffice to obtain an ϵ-accurate solution. Thus, the complexity bound (17) is within constant factors of the lower bound (14).

Digression: A faster method?

A curious multistep algorithm is the nonmonotonic gradient descent of Barzilai and Borwein [1988] (BB). Here one first computes the differences $\boldsymbol{s}_k = \boldsymbol{x}_k - \boldsymbol{x}_{k-1}$, $\boldsymbol{y}_k = \nabla\phi(\boldsymbol{x}_k) - \nabla\phi(\boldsymbol{x}_{k-1})$, and then iterates (5) using the stepsize

$$\alpha_k = \langle \boldsymbol{s}^k, \boldsymbol{y}^k\rangle / \|\boldsymbol{y}^k\|^2, \quad \text{or} \quad \alpha_k = \|\boldsymbol{s}^k\|^2 / \langle \boldsymbol{s}^k, \boldsymbol{y}^k\rangle. \tag{18}$$

These stepsizes may be viewed as scalar approximations to the Hessian or its inverse; this approximation of the curvature explains to some extent the impressive empirical speedups enjoyed by BB gradient-descent (see Fig. 2).

3.1.1 Constrained problems

We can accommodate 'simple' constraints into the above gradient based methods without changing their upper complexity bounds. In particular, say $\boldsymbol{x}$ lies in $\mathcal{X}$ which is "simple" so that the *orthogonal projection* (6) can be computed *exactly*. The upper and lower bounds stated in Theorems 3.4 and 3.8 extend easily to this *constrained* problem.

The original proofs extend easily because we can exploit the fact that $P_{\mathcal{X}}$ is *firmly nonexpansive*:

$$\langle P_{\mathcal{X}}\boldsymbol{x} - P_{\mathcal{X}}\boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y}\rangle \leq \|\boldsymbol{x} - \boldsymbol{y}\|^2. \tag{19}$$

The details are an instructive, so we leave them as an exercise for the reader. Nesterov's optimal method (16) too, extends easily if we use the projected update $\boldsymbol{x}_{k+1} = P_{\mathcal{X}}\big(\boldsymbol{y}_k - \frac{1}{L}\nabla\phi(\boldsymbol{y}_k)\big)$.

But what happens if the projection cannot be computed exactly? Indeed, frequently the constraint set is not so simple and the projection $P_{\mathcal{X}}$ can be computed only approximately. In this case, substantial changes to the algorithm and its convergence analysis must be made. For an accessible account, we refer the reader to the recent work of Schmidt et al. [2011], who discuss how to handle inexact projections (and proximity operators, to be introduced in Section 4.3).

4 Nonsmooth Convex Optimization

After our brief discussion of smooth convex optimization we now turn to a class of problems pervasive in machine learning: *nonsmooth convex optimization*. More specifically, we consider optimizing convex functions that are Lipschitz continuous over a compact set containing the solution.

Example 4.1 (Hinge loss SVM)**.** Let $\{(\boldsymbol{x}_i, y_i) : 1 \leq i \leq m\}$ be labeled training examples, where $\boldsymbol{x}_i \in \mathbb{R}^n$ are feature vectors and $y_i \in \{\pm 1\}$ their labels. One way to classify an unlabeled point $\boldsymbol{x}$ is to assign to it the label $\operatorname{sgn}(\boldsymbol{w}^T\boldsymbol{x} + b)$, where $\boldsymbol{w} \in \mathbb{R}^n$ is a weight vector that must be learned and $b \in \mathbb{R}$ is a "bias" term. A fundamental choice for computing $\boldsymbol{w}$ and b is the *Hinge loss SVM*, which involves solving the nonsmooth problem:

$$\min_{\boldsymbol{w},b} \ \phi(\boldsymbol{w}, b) := \tfrac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum\nolimits_{i=1}^{m} \max(0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)), \tag{20}$$

where $C > 0$ is a tradeoff parameter. The hinge loss terms in (20) are easily seen to be Lipschitz continuous and convex. Assuming that the optimal $\boldsymbol{w}^*$ lies in a ball of radius D, we see that $2D$ suffices as a Lipschitz constant for the (convex) regularizer $\|\boldsymbol{w}\|_2^2$. Thus, $\phi(\boldsymbol{w}, b)$ is Lipschitz continuous and convex.

4.1 Basic Theory

We start our discussion of nonsmooth convex optimization by recalling an extremely important concept from convex analysis.

Definition 4.2 (Subdifferential)**.** Let ϕ be convex. For any $\boldsymbol{y} \in \operatorname{dom}\phi$ the *subdifferential* of ϕ at $\boldsymbol{y}$, written $\partial\phi(\boldsymbol{y})$, is defined to be the set

$$\partial\phi(\boldsymbol{y}) := \{\boldsymbol{g} \mid \phi(\boldsymbol{x}) \geq \phi(\boldsymbol{y}) + \langle \boldsymbol{g}, \boldsymbol{x} - \boldsymbol{y}\rangle \;\; \forall \boldsymbol{x} \in \operatorname{dom}\phi\}\,. \tag{21}$$

The subdifferential generalizes the concept of a gradient: if ϕ is differentiable, then the subdifferential is a singleton $\partial\phi(\boldsymbol{y}) = \{\nabla\phi(\boldsymbol{y})\}$. Akin to gradients, subdifferentials also help characterize optimality. Indeed, using (21) an easy exercise shows that for a given point $\boldsymbol{x}^*$,

$$\phi(\boldsymbol{x}^*) \leq \phi(\boldsymbol{x}) \quad \text{for all } \boldsymbol{x}, \quad \text{if and only if } \;\; 0 \in \partial\phi(\boldsymbol{x}^*). \tag{22}$$

Given these properties of the subdifferential, we may hope to generalize iteration (5) to accommodate nondifferentiable convex functions. A fairly natural choice is to introduce the (projected) *subgradient method*

$$\boldsymbol{x}_{k+1} = P_{\mathcal{X}}(\boldsymbol{x}_k - \alpha_k \boldsymbol{g}_k), \quad k = 0, 1, \ldots, \tag{23}$$

where, instead of the gradient we use an arbitrary *subgradient* $\boldsymbol{g}_k \in \partial\phi(\boldsymbol{x}_k)$; α_k is a suitable stepsize. Although similar in form to (5), iteration (23) differs crucially: unlike the (negative) gradient, an arbitrary subgradient does not provide a direction of descent. This limitation forces us to use stepsizes that are less flexible than for gradient-descent, and also contributes to making (23) converge slowly.

Example 4.3 (Nesterov [2008])**.** Let $\phi(x) = |x|$, for $x \in \mathbb{R}$. The subgradient method (23) generates $x_{k+1} = x_k - \alpha_k g_k$, where $g_k \in \partial|x_k|$. If $x_0 = 1$ and $\alpha_k = \frac{1}{\sqrt{k+1}} + \frac{1}{\sqrt{k+2}}$ (this stepsize is known to be optimal [Nesterov, 2008]), then $|x_k| = \frac{1}{\sqrt{k+1}}$, so that $O(\frac{1}{\epsilon^2})$ iterations are needed to obtain ϵ-accuracy.

The dismal behavior shown in Example 4.3 is typical for the subgradient method which exhibits $O(1/\sqrt{k})$ convergence. One may wonder: *Can we do better than the subgradient method?* The answer is, unfortunately, '*No.*' Theorem 4.4 asserts this claim formally.

Theorem 4.4 (Nesterov [2004])**.** *Assume that the solution $\boldsymbol{x}^*$ lies in a Euclidean ball of radius $D > 0$ around a given initial point $\boldsymbol{x}_0$, denoted $\mathcal{B}$. There exists a function ϕ in $C^0_L(\mathcal{B})$ (with $L > 0$), such that for $0 \leq k \leq n-1$, the inequality*

$$\phi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}^*) \geq \tfrac{LD}{2(1+\sqrt{k+1})}, \tag{24}$$

holds for any algorithm that generates $\boldsymbol{x}_k$ by linearly combining the previous iterates and subgradients (cf. (10)).

Compared with the C^1_L convex case, the situation for the nonsmooth case looks much worse. Nesterov [2003, 2005a] brought good news by showing how to circumvent the $O(1/\sqrt{k})$ barrier of (24) by making a simple (in hindsight) but far-reaching observation: *we don't always work with black-boxes but explicitly know the problem structure, which can be exploited to obtain faster algorithms.* Let us now describe the key consequences of this observation.

4.2 Smooth Minimization of Nonsmooth Functions

Intuitively, if we could replace a nonsmooth problem with a "good enough" smooth approximation, we could approximately solve the nonsmooth problem faster. This idea may be expected to work because on a compact set, a Lipschitz continuous convex function can be approximated to any uniform accuracy $\epsilon > 0$ by a C^1_L convex function. If L is of the order $O(1/\epsilon)$, then an optimal method that requires $O(\sqrt{L/\epsilon})$ iterations could yield a nonsmooth optimization method that converges as $O(1/\epsilon)$, thereby breaking the $O(1/\sqrt{k})$ barrier (which corresponds to $O(1/\epsilon^2)$).

Nesterov [2005a] builds on this intuition to consider nonsmooth functions $\phi(\boldsymbol{x})$ that possess an *explicit max-structure*, that is,

$$\phi(\boldsymbol{x}) = \hat{\phi}(\boldsymbol{x}) + \max_{\boldsymbol{z}\in\mathcal{Z}\subseteq\mathbb{R}^m}\{\langle \boldsymbol{A}\boldsymbol{x},\, \boldsymbol{z}\rangle - \hat{\phi}(\boldsymbol{z})\}, \tag{25}$$

where $\hat{\phi}$ is convex and lies in $C^1_{L(\hat{\phi})}(\mathcal{X})$; $\hat{\phi}(\boldsymbol{z})$ is continuous and convex on $\mathcal{Z}$, and $\boldsymbol{A} \in \mathbb{R}^{m\times n}$. For such ϕ, he then introduces a smoothness parameter $\mu > 0$, and defines the *smooth approximation:*[4]

$$\phi_\mu(\boldsymbol{x}) := \hat{\phi}(\boldsymbol{x}) + \max_{\boldsymbol{z}\in\mathcal{Z}}\{\langle \boldsymbol{A}\boldsymbol{x},\, \boldsymbol{z}\rangle - \hat{\phi}(\boldsymbol{z}) - \tfrac{1}{2}\mu\|\boldsymbol{z}\|^2\}. \tag{26}$$

The 'max' term in (26) has a unique solution (since $\hat{\phi}$ is continuous, $\|\boldsymbol{z}\|^2$ is strongly convex and $\mathcal{Z}$ is compact), say $\boldsymbol{z}_\mu(\boldsymbol{x})$. From standard results [e.g., Rockafellar and Wets, 1998, Theorem 2.26] it follows that

$$\nabla\phi_\mu(\boldsymbol{x}) = \nabla\hat{\phi}(\boldsymbol{x}) + \boldsymbol{A}^T\boldsymbol{z}_\mu(\boldsymbol{x}), \tag{27}$$

with a Lipschitz constant given by

$$L(\phi_\mu) := L(\hat{\phi}) + \tfrac{1}{\mu}\|\boldsymbol{A}\|^2. \tag{28}$$

Depending on the set $\mathcal{Z}$, instead of $\|\boldsymbol{z}\|^2$, a different "prox-function" might be more appropriate—we omit discussion for simplicity and refer the reader to [Juditsky and Nemirovski, 2011a, Nesterov, 2005a]. Instead, we directly present Nesterov's smoothing algorithm.

For $k \geq 0$ iterate:

$$\left\lfloor \begin{array}{l} \boldsymbol{y}_k = P_{\mathcal{X}}(\boldsymbol{x}_k - \frac{1}{L(\phi_\mu)}\nabla\phi_\mu(\boldsymbol{x}_k)) \\ \boldsymbol{z}_k = \operatorname{argmin}_{\boldsymbol{x}\in\mathcal{X}} \frac{1}{2}L(\phi_\mu)\|\boldsymbol{x}\|^2 + \sum_{i=0}^{k} \frac{i+1}{2}\langle\nabla\phi_\mu(\boldsymbol{x}_i),\, \boldsymbol{x}\rangle \\ \boldsymbol{x}_{k+1} = \frac{2}{k+3}\boldsymbol{z}_k + \frac{k+1}{k+3}\boldsymbol{y}_k \end{array} \right. \tag{29}$$

To describe the complexity of iteration (29), we need the (primal and dual) diameters $D_{\mathcal{X}} := \max_{\boldsymbol{x}\in\mathcal{X}}\|\boldsymbol{x}\|$ and $D_{\mathcal{Z}} := \max_{\boldsymbol{z}\in\mathcal{Z}}\|\boldsymbol{z}\|$. Nesterov [2005a, Thm. 3] shows that upon setting $\mu = \frac{2\|\boldsymbol{A}\|}{N+1}\frac{D_{\mathcal{X}}}{D_{\mathcal{Z}}}$, after N iterations of (29), the vector $\boldsymbol{y}_N$ solves (25) to an accuracy upper-bounded by

$$\frac{4\|\boldsymbol{A}\|}{N+1}D_{\mathcal{X}}D_{\mathcal{Z}} + \frac{4L(\hat{\phi})D^2_{\mathcal{X}}}{(N+1)^2}. \tag{30}$$

And what's more, it can be shown that (modulo constant factors) this bound is unimprovable [Juditsky and Nemirovski, 2011a].

An instructive example where algorithm (29) applies is shown as Example 4.5 below.

[4] This approximation is essentially the negative *Moreau envelope* of $\hat{\phi}$.

Example 4.5 (Least absolute deviations)**.** A regression task that is more robust (perchance less stable) than least-squares is least absolute deviations (LAD) [Portnoy and Koenker, 1997]. Here, we minimize the ℓ_1-norm cost $\phi(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_1$. Unlike least-squares, LAD does not have an analytic solution, so it has attracted several iterative approaches; we are, however, unaware if Nesterov's smoothing technique has been applied to it so far.
Since the cost ϕ is separable, on noting $|x_i| = \max_{|z_i|\leq 1} z_i x_i$, we obtain

$$\phi(\boldsymbol{x}) = \max_{\|\boldsymbol{z}\|_\infty \leq 1}\{\langle \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b},\, \boldsymbol{z}\rangle\}, \qquad \text{(notice: } \hat{\phi} = 0\text{)},$$

so that $\phi_\mu(\boldsymbol{x}) := \max_{\|\boldsymbol{z}\|_\infty \leq 1}\{\langle \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b},\, \boldsymbol{z}\rangle - \frac{1}{2}\mu\|\boldsymbol{z}\|^2\}$. Hence, $\boldsymbol{z}_\mu(\boldsymbol{x})$ can be computed simply by projecting the scaled residual $\mu^{-1}(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b})$ onto the ball $\|\boldsymbol{z}\|_\infty \leq 1$. Notice that $\|\boldsymbol{z}\| \leq \sqrt{n}\|\boldsymbol{z}\|_\infty \leq \sqrt{n} =: D_{\mathcal{Z}}$. Some linear algebra shows that we can essentially restrict $\boldsymbol{x}$ to lie in a Euclidean ball with diameter $D_{\mathcal{X}} < \infty$. Thus, we can invoke (29) to solve LAD.

For more sophisticated examples, see [Ying et al., 2009], who apply the smoothing technique to *metric learning*, and [Stobbe and Krause, 2010], who apply the smoothing technique to approximately minimize a subclass of decomposable submodular functions.

4.2.1 Excessive Gap Technique

Scheme (29) has a disadvantage: to set the smoothing parameter μ, the number of steps must be known in advance. Nesterov [2005b] overcomes this disadvantage by assuming slightly more structure, for which he develops a new *excessive-gap technique* (EGT). This leads to $O(1/k)$ rate for problems with max-structure, and a faster $O(1/k^2)$ rate for strongly convex objectives. We summarize the key ideas below.

First, some convex analysis shows that the (Fenchel) *dual* to (25) is

$$g(\boldsymbol{z}) := -\hat{\phi}(\boldsymbol{z}) + \min_{\boldsymbol{x}\in\mathcal{X}}\{\langle \boldsymbol{A}\boldsymbol{x},\, \boldsymbol{z}\rangle + \hat{\phi}(\boldsymbol{x})\}. \tag{31}$$

For this dual, we let $\nu > 0$ and introduce the smooth approximation

$$g_\nu(\boldsymbol{z}) := -\hat{\phi}(\boldsymbol{z}) + \min_{\boldsymbol{x}\in\mathcal{X}}\{\langle \boldsymbol{A}\boldsymbol{x},\, \boldsymbol{z}\rangle + \hat{\phi}(\boldsymbol{x}) + \tfrac{1}{2}\nu\|\boldsymbol{x}\|^2\}. \tag{32}$$

We require the gradient of (32), which is simply given by

$$\nabla g_\nu(\boldsymbol{z}) = -\nabla\hat{\phi}(\boldsymbol{z}) + \boldsymbol{A}\boldsymbol{x}_\nu(\boldsymbol{z}),$$

where $\boldsymbol{x}_\nu(\boldsymbol{z})$ denotes the unique solution to the 'min' in (32). One may quickly verify that the gradient ∇g_ν is Lipschitz with the constant

$$L(g_\nu) := L(\hat{\phi}) + \tfrac{1}{\nu}\|\boldsymbol{A}\|^2.$$

Observe that *weak-duality* implies that $g(\boldsymbol{z}) \leq \phi(\boldsymbol{x})$ for all pairs $(\boldsymbol{x}, \boldsymbol{z}) \in \mathcal{X} \times \mathcal{Z}$, while *strong-duality* mandates $\min_{\boldsymbol{x}\in\mathcal{X}} \phi(\boldsymbol{x}) = \max_{\boldsymbol{z}\in\mathcal{Z}} g(\boldsymbol{z})$. Thus, for a given pair $(\boldsymbol{x}, \boldsymbol{z})$ we may compute the *duality gap* $\phi(\boldsymbol{x}) - g(\boldsymbol{z})$, but what about the "smoothed" gap $\phi_\mu(\boldsymbol{x}) - g_\nu(\boldsymbol{z})$? This gap can be positive or negative (as any of the inequalities $g \leq \phi$, $\phi_\mu \leq \phi$, or $g \leq g_\nu$ can hold), so it is useful to consider the *excessive gap condition*:

$$\phi_\mu(\bar{\boldsymbol{x}}) \leq g_\nu(\bar{\boldsymbol{z}}) \quad \text{for a pair} \quad (\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}) \in \mathcal{X} \times \mathcal{Z}. \tag{33}$$

Condition (33) lies at the heart of the EGT. To see why, observe that

$$\phi(\boldsymbol{x}) - \mu D_{\mathcal{Z}}^2 \leq \phi_\mu(\boldsymbol{x}) \quad \text{and} \quad g(\boldsymbol{z}) + \nu D_{\mathcal{X}}^2 \geq g_\nu(\boldsymbol{z}). \tag{34}$$

Thus, for a primal-dual pair $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}})$ that satisfies the excessive gap condition (33), the corresponding duality gap is bounded by

$$\phi(\bar{\boldsymbol{x}}) - g(\bar{\boldsymbol{z}}) \leq \mu D_{\mathcal{Z}}^2 + \nu D_{\mathcal{X}}^2. \tag{35}$$

In other words, the duality gap is bounded by a function linear in μ and ν. So, if both of these parameters shrink to zero, the duality gap will also shrink to zero, at least as fast.

The aim of Nesterov's EGT is, therefore, to generate a sequence $\{(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{z}}_k)\}$ of primal-dual iterates that satisfy condition (33), and to simultaneously choose scalar sequences $\{\mu_k\}$ and $\{\nu_k\}$ that tend to zero at the desired rate. For example, when $L(\hat{\phi}) = L(\hat{\phi}) = 0$, then Nesterov [2005b, Iteration (6.8)] shows how to generate such sequences, and to thereby obtain the desired $O(1/k)$ upper-bound

$$\phi(\bar{\boldsymbol{x}}_k) - g(\bar{\boldsymbol{z}}_k) \leq \frac{4\|\boldsymbol{A}\|}{k+1} D_{\mathcal{X}} D_{\mathcal{Z}}. \tag{36}$$

If in addition the function $\hat{\phi}$ in (25) is σ-strongly-convex, then an even faster rate of convergence is possible; specifically, it holds that

$$\phi(\bar{\boldsymbol{x}}_k) - g(\bar{\boldsymbol{z}}_k) \leq \frac{4D_{\mathcal{Z}}(L(\hat{\phi}) + \sigma^{-1}\|\boldsymbol{A}\|^2)}{(k+1)(k+2)}. \tag{37}$$

A point worth noting about (37) is that in contrast to (36), the rate does not depend on $D_{\mathcal{X}}$, so in principle, the set $\mathcal{X}$ need not be bounded.

Example 4.6 (Structured prediction)**.** In [Zhang et al., 2011] the authors apply the EGT (with non-Euclidean prox-functions) to maximum-margin Markov networks, by minimizing the regularized 'hinge-loss':

$$\phi(\boldsymbol{w}) := \frac{\gamma}{2}\|\boldsymbol{w}\|^2 + \frac{1}{m}\sum\nolimits_{i=1}^{m} \max_{\boldsymbol{y}\in\mathcal{Y}} \phi_i(\boldsymbol{y}, \boldsymbol{w}; \boldsymbol{z}_i),$$

where $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$ are labeled data, $\mathcal{Y}$ the labels space, and ϕ_i are loss functions. Observe that the same ideas also applies to the hinge-loss SVM of Example 4.1 and other regularized prediction problems.

Example 4.7 (Large-scale Linear Programming)**.** Chen and Burer [2011] apply the EGT (with some modifications) to solve large-scale instances of the following linear programming problem

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y}} \quad & \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{y} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \leq \boldsymbol{y}, \qquad \boldsymbol{x}, \boldsymbol{y} \geq 0. \end{aligned}$$

The authors also show applications to some machine learning problems.

Note. The smoothing ideas can be generalized to problems whose objective function ϕ admits the *saddle-point* representation

$$\phi(\boldsymbol{x}) := \max\nolimits_{\boldsymbol{z}\in\mathcal{Z}} \quad \Phi(\boldsymbol{x}, \boldsymbol{z}), \tag{38}$$

where Φ has a Lipschitz continuous gradient on $\mathcal{X} \times \mathcal{Z}$. Instead of Nesterov's smoothing algorithms, we can also solve (38) by using the Mirror-Prox algorithm [Nemirovski, 2004] that yields a $O(1/k)$ rate.

4.3 Composite objective minimization

We just saw how exploitation of structure can be used to circumvent lower bounds and improve convergence rates substantially. Can this idea be taken even further? It turns out that for certain problems, it indeed can. Assume that the objective can be written as

$$\phi(\boldsymbol{x}) := \ell(\boldsymbol{x}) + r(\boldsymbol{x}), \tag{39}$$

where $\ell \in C_L^1$ and r is a continuous convex regularizer. Objective (39) is called *composite* as it is a mix of smooth and nonsmooth terms.[5]

Composite objectives enjoy great importance in machine learning, largely because of their flexibility and wide applicability. The smooth part measures quality of the estimation, while the nonsmooth part encourages the solution to satisfy certain desirable structural properties such as sparsity. We note in passing that one can alternatively view the loss $\ell(\boldsymbol{x})$ as a negative log-likelihood term and the regularizer $r(\boldsymbol{x})$ as a prior, though we do not pursue such connections further.

[5]This formulation is complementary to (25) where the max-term may be viewed as a nonsmooth loss and $\hat{\phi}$ as a regularizer.

Example 4.8. Let $\ell(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2$ denote a generic quadratic loss; we list some corresponding choices for the regularizer $r(\boldsymbol{x})$ below.

- *Lasso* [Tibshirani, 1996] uses $r(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_1$, primarily to effect feature selection. The ℓ_1-norm is known to prefer sparse solutions (i.e., solutions with many zeros), which effectively selects columns of matrix $\boldsymbol{A}$ that play a dominant role in explaining the observation vector $\boldsymbol{b}$. The ℓ_1-norm also enjoys great popularity in the field of compressed sensing, where it is formally used as a proxy for the cardinality function [Baraniuk, 2007].

- *Fused lasso* [Tibshirani et al., 2005] combines the ℓ_1-norm on the vector $\boldsymbol{x}$ with an ℓ_1-norm on the "discrete gradient" of $\boldsymbol{x}$, i.e., it uses $r(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_1 + \sum_{i=1}^{n-1}|x_{i+1} - x_i|$. This combination elicits not only sparse vectors $\boldsymbol{x}$, but also enforces their gradients to be sparse—thus, a few large changes will be preferred to several small changes in adjacent components of $\boldsymbol{x}$. This idea proves useful in applications such as change detection or even in feature selection [Tibshirani et al., 2005].

- *Total variation.* Suppose the variable $\boldsymbol{x}$ is a matrix in $\mathbb{R}^{n\times n}$. The classical total-variation regularizer used for image denoising is [Rudin et al., 1992]:

$$r(\boldsymbol{x}) = \sum\nolimits_{1\le i,j<n} \|(\nabla x)_{i,j}\|_2, \text{ where } (\nabla x)_{i,j} = \begin{bmatrix} x_{i+1,j} - x_{i,j} \\ x_{i,j+1} - x_{i,j} \end{bmatrix}.$$

  However, we alert the reader in passing that this choice of $r(\boldsymbol{x})$ does not full fit with the theory that we will describe next, though it can be accommodated using the "inexact" methods proposed in [Schmidt et al., 2011].

- *Mixed-norm regression.* Assume that $\boldsymbol{x}$ is split into potentially overlapping "groups" of subvectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_G$. Instead of selecting features individually, as is done in Lasso, we could now select features groupwise, e.g., using

$$r(\boldsymbol{x}) := \left(\sum\nolimits_{i=1}^{G} \|\boldsymbol{x}_i\|_q^p\right)^{1/p}.$$

We refer the reader to [Bach et al., 2011] for a more detailed development.

Let us now see how to solve (39). Nesterov [2007] showed how to minimize ϕ at a rate limited by only the smooth part of the objective as long as r is *simple*; that is, the following operator is assumed to be computable exactly.[6]

Definition 4.9 (Proximity operator)**.** Let $r : \mathcal{X} \subseteq \mathbb{R}^n \to \mathbb{R}$ be lower semicontinuous and convex. The *proximity operator* for r, indexed by scalar $\eta > 0$, is the following nonlinear map:

$$\text{prox}_\eta^r \;:\quad \boldsymbol{y} \mapsto \operatorname{argmin}_{\boldsymbol{x}\in\mathcal{X}} \left(r(\boldsymbol{x}) + \tfrac{1}{2\eta}\|\boldsymbol{x} - \boldsymbol{y}\|^2\right). \tag{40}$$

Operator (40) generalizes the projection operator (*cf.* (6)): if $r(\boldsymbol{x})$ is the indicator function of set $\mathcal{X}$, then $\text{prox}_\eta^r \equiv P_{\mathcal{X}}$. One of its most useful properties is its nonexpansivity, shown as Lemma 4.1.

[6] Large parts of the theory also carry over for inexact proximity operators, see [Schmidt et al., 2011] for a nice account.

Lemma 4.1 (Nonexpansivity [see e.g., Combettes and Pesquet, 2010, Lemma 2.4])**.** *The operator* prox_η^r *satisfies*

$$\|\text{prox}_\eta^r x - \text{prox}_\eta^r y\| \leq \|x - y\|, \quad \forall\ x, y \in \mathbb{R}^n. \tag{41}$$

This operator enjoys several other useful properties, and we encourage the reader to read the excellent survey [Combettes and Pesquet, 2010] for more information.

In analogy to iteration (5) where the projection operator $P_\mathcal{X}$ handles constraints, we may wonder similarly whether prox_η^r allows us to handle the nonsmooth term $r(\boldsymbol{x})$, if we simply iterate

$$\boldsymbol{x}_{k+1} = \text{prox}_{\eta_k}^r(\boldsymbol{x}_k - \alpha_k \nabla \ell(\boldsymbol{x}_k)), \quad k = 0, 1, \ldots. \tag{42}$$

It turns out that this is indeed possible. Iteration (42) is the so-called *forward backward splitting* [e.g., Combettes and Wajs, 2005] method, which converges at the rate $O(1/k)$, matching the rate achieved by the smoothing methods of the previous section. Extending the analogy to gradient-projection, it is therefore fitting to ask whether this $O(1/k)$ rate can be improved?

By now, no surprise: such an improved version was developed by Nesterov [2007]; also by Beck and Teboulle [2009]. These improved methods achieve the optimal rate $O(1/k^2)$, and if in addition ℓ is strongly convex, their rate improves to (optimal) linear [Nesterov, 2007]. We summarize Beck and Teboulle's method FISTA, which incidentally is almost implicit (modulo the prox-operator) in Nesterov's optimal method (16).

Choose $\boldsymbol{x}_0 \in \mathbb{R}^n$; $\boldsymbol{y}_0 = \boldsymbol{x}_0$; and $\alpha_0 = 1$
For $k \geq 0$ iterate:

$$\left\lfloor \begin{array}{ll} \boldsymbol{x}_{k+1} & = \text{prox}_{1/L}^r(\boldsymbol{y}_k - \frac{1}{L}\nabla \ell(\boldsymbol{y}_k)) \\ \alpha_{k+1} & = (1 + \sqrt{4\alpha_k^2 + 1})/2 \\ \boldsymbol{y}_{k+1} & = \boldsymbol{x}_{k+1} + \frac{\alpha_k - 1}{\alpha_{k+1}}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) \end{array} \right. \tag{43}$$

From (43) it is clear that the computing $\boldsymbol{x}_{k+1}$ is the hard part. Thus, a lot of attention has been paid to efficient computation of proximity operators. Discussing this computation would take us beyond the scope of this chapter; we refer the interested reader to [Sra et al., 2011, Liu et al., 2009]; also see the useful survey by [Patriksson, 2005].

Another digression: a faster method?

We saw that among gradient-based methods, the nonmonotonic method of Barzilai and Borwein [1988] can empirically outperforms gradient descent and Nesterov's optimal method (Fig. 2). Do we have a similarly strong method for the composite objective case? There are at least two such methods: SpaRSA [Wright et al., 2009] and TRIP [Kim et al., 2010]. The latter is based on a nonsmooth trust-region strategy that builds quadratic approximations to its objective function by using BB-formulae, and then uses proximity operators to handle the nonsmooth part; the trust-region strategy ensures convergence. Empirically, TRIP seems to yield impressive performance—see Figs. 3 and 4.

5 Stochastic optimization

So far we assumed access to a noiseless oracle. It is more realistic to consider *noisy oracles*, where one does not have access to exact objective function or gradient values, but rather to their noisy estimates (usually zero mean, bounded variance noise is assumed).

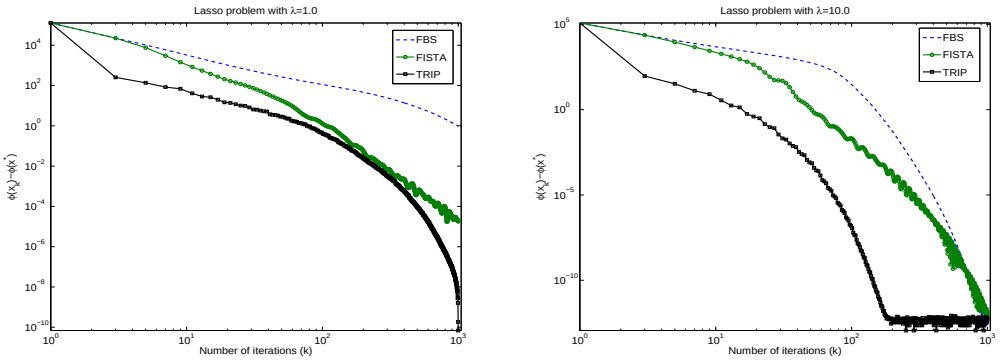

Figure 3: Lasso problem that solves $\min \frac{1}{2}\|\boldsymbol{D}^T\boldsymbol{x} - \boldsymbol{b}\|^2 + \lambda\|\boldsymbol{x}\|_1$, where $\boldsymbol{D}$ is as in Fig. 2 with $n = 500$. Left plot shows a run with $\lambda = 1$, which leads to $\approx 7\%$ sparsity; right plot shows run with $\lambda = 10$, which leads to $\approx 41\%$ sparsity. As sparsity increases, FBS performs similar to FISTA. Surprisingly, the TRIP method of Kim et al. [2010], which combines Barzilai-Borwein stepsizes with proximity operators, outperforms both, more so for higher sparsity.

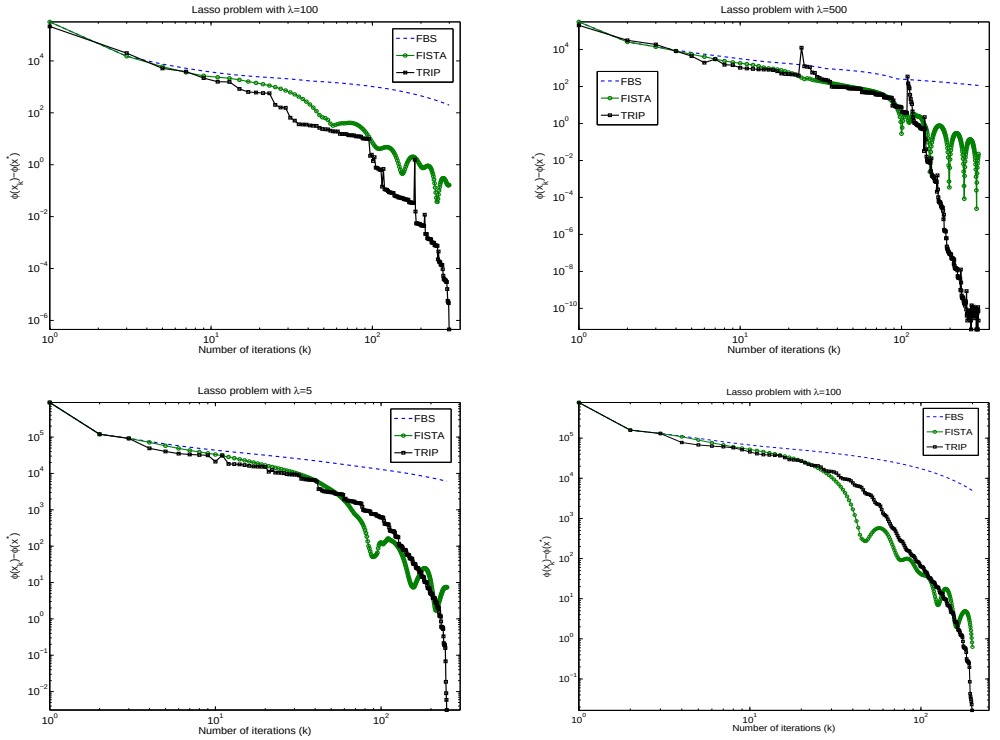

Figure 4: Lasso problem on two UCI datasets. Top row: Abalone dataset (4177×8); bottom row: Year Prediction dataset (463715×90) [Frank and Asuncion, 2010]. Both FISTA and TRIP outperform FBS, and TRIP seems to yield higher accuracy solutions. Note that we used datasets with more rows than columns to make it easy to compute the optimal L and μ parameters needed by FISTA and FBS.

This setup is fundamental to machine learning and has attracted enormous interest in the community. Therefore, it is impossible to survey this topic here, and we refer the reader to the tutorial [Srebro and Tewari, 2010] that expostulates the connection between stochastic optimization and machine learning; see also [Bottou and Bousquet, 2011] and [Ghadimi and Lan, 2010] for background and algorithms. We limit our attention to the very recent work of Ghadimi and Lan [2010], because of its generality and algorithmic simplicity.

In this section too, we are interested in computing

$$\phi^* := \min_{\boldsymbol{x}\in\mathcal{X}} \big(\phi(\boldsymbol{x}) := f(\boldsymbol{x}) + r(\boldsymbol{x})\big), \tag{44}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ is a compact convex set and $r(\boldsymbol{x})$ is a 'simple' convex function as before. We assume that $f : \mathcal{X} \to \mathbb{R}$ is convex and satisfies

$$\begin{aligned}\tfrac{1}{2}\mu\|\boldsymbol{y}-\boldsymbol{x}\|^2 \le f(\boldsymbol{y}) - f(\boldsymbol{x}) - \langle f'(\boldsymbol{x}),\, \boldsymbol{y}-\boldsymbol{x}\rangle \\ \le \tfrac{1}{2}L\|\boldsymbol{y}-\boldsymbol{x}\|^2 + M\|\boldsymbol{y}-\boldsymbol{x}\|, \quad \text{for all } \boldsymbol{x},\boldsymbol{y}\in\mathcal{X},\end{aligned} \tag{45}$$

but that it is *not* known exactly. Specifically, assume that f (and f') are available to us via a stochastic oracle, which on receiving $\boldsymbol{x}_k$ as input, returns the noisy estimates ($\{\xi_k\}$ is a sequence of i.i.d. noise variables):

$$F(\boldsymbol{x}_k,\xi_k) \in \mathbb{R}, \quad \text{and } G(\boldsymbol{x}_k,\xi_k) \in \mathbb{R}^n. \tag{46}$$

Further, it is assumed that the estimates (46) satisfy

$$\mathbb{E}[F(\boldsymbol{x},\xi_k)] = f(\boldsymbol{x}), \quad \mathbb{E}[G(\boldsymbol{x},\xi_k)] = f'(\boldsymbol{x}) \in \partial f(\boldsymbol{x}), \tag{47}$$

and that the variance of the (sub)gradient estimates is bounded, so that

$$\mathbb{E}[\|G(\boldsymbol{x},\xi_k) - f'(\boldsymbol{x})\|^2] \le \sigma^2 \quad \text{for } k \ge 1. \tag{48}$$

The above setup is parameterized by the tuple (L, M, μ, σ), different choices of which cover different scenarios. For example, $\sigma = 0$ reduces to the zero-noise, non-stochastic case. If $f \in S^1_{L,\mu}$, then $M = 0$, while if $f \in C^0_{L(f)}$, then (45) holds with $L = 0$, $\mu = 0$, and $M = 2L(f)$. Ghadimi and Lan [2010] present a general algorithm for solving the (L, M, μ, σ) setup; a simplified version of which is presented below.

Let $\boldsymbol{x}_0 \in \mathcal{X}$; $\boldsymbol{z}_0 = \boldsymbol{x}_0$; and $\gamma \ge 0$ be provided.
For $k \ge 1$ iterate:

$$\alpha_k = \tfrac{2}{k+1}; \quad \gamma_k = \tfrac{4L}{k(k+1)} + \tfrac{2\gamma}{\sqrt{k}}; \quad \tau_k = \tfrac{(1-\alpha_k)(\mu+\gamma_k)}{\gamma_k + (1-\alpha_k^2)\mu}$$

$$\boldsymbol{y}_k = \tau_k \boldsymbol{z}_{k-1} + (1-\tau_k)\boldsymbol{x}_{k-1}$$

Invoke stochastic oracle to compute $\boldsymbol{g}_k = G(\boldsymbol{y}_k, \xi_k)$

$$\left\lfloor \begin{array}{ll} \hat{\boldsymbol{x}}_k & = \alpha_k\mu\boldsymbol{y}_k + ((1-\alpha_k)\mu + \gamma_k)\boldsymbol{x}_{k-1} - \alpha_k\boldsymbol{g}_k \\ \boldsymbol{x}_k & = \operatorname{prox}^r_{\alpha_k/(\gamma_k+\mu)}(\hat{\boldsymbol{x}}_k/(\mu+\gamma_k)) \\ \boldsymbol{z}_k & = \alpha_k\boldsymbol{x}_k + (1-\alpha_k)\boldsymbol{z}_{k-1}; \end{array} \right. \tag{49}$$

The above algorithm enjoys the following complexity results.

Theorem 5.1 (Ghadimi and Lan [2010]). *Let $\{\boldsymbol{z}_k\}_{k\geq 1}$ be generated by (49) and let $D := \sup_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}^*\|$, where $\phi(\boldsymbol{x}^*) = \phi^*$. If we set $\gamma \approx 1.94\sqrt{M^2+\sigma^2}/D$, then for the case with $\mu = 0$ we have*

$$\mathbb{E}[\phi(\boldsymbol{z}_k) - \phi^*] \leq \frac{2LD^2}{k(k+1)} + \frac{34D\sqrt{M^2+\sigma^2}}{5\sqrt{k}}. \tag{50}$$

For $\mu > 0$ we can set $\gamma = 0$; then we have

$$\mathbb{E}[\phi(\boldsymbol{z}_k) - \phi^*] \leq \frac{2LD^2}{k(k+1)} + \frac{8(M^2+\sigma^2)}{\mu(k+1)}. \tag{51}$$

We omit the proof as it is very tedious. It is worth noting that in the absence of strong convexity (i.e., $\mu = 0$), the result (50) is optimal to within constants. For strongly convex optimization, the result (51) is optimal in the slower $O(1/k)$ term, but suboptimal in the faster term. Specifically, it is known [Nemirovsky and Yudin, 1983] that to find a point $\bar{\boldsymbol{x}} \in \mathcal{X}$ such that $\mathbb{E}[\phi(\bar{\boldsymbol{x}}) - \phi^*] \leq \epsilon$, the number of calls to the stochastic oracle cannot be smaller than

$$C\Big(\sqrt{\tfrac{L}{\mu}}\log\big(\tfrac{LD^2}{\epsilon}\big) + \tfrac{(M+\sigma)^2}{\mu\epsilon}\Big),$$

where D is as defined in Theorem 5.1, and C is an absolute constant.

Ghadimi and Lan [2010] also develop a multistage version of (49) that achieves the above cited lower bound, though at the cost of a more complicated algorithm; large-deviation results that show bounds in terms of probability (rather than expected values) are also presented.

5.1 Other closely related algorithms

We conclude this section by mentioning two other setups algorithmically related to stochastic optimization: (i) online convex optimization; and (ii) incremental (sub)gradient and proximal methods. For an accessible account of the former please see [Hazan, 2011], while for a survey treatment of the latter see [Bertsekas, 2010].

In the stochastic setup we assume the 'samples' ξ_i to be i.i.d. In contrast, in online optimization, we receive samples that need not follow any distribution and may even be chosen adversarially. In general, at each step an online algorithm selects a point, say $\boldsymbol{x}_k \in \mathcal{X}$ (from a fixed set $\mathcal{X}$). Then, the adversary reveals its cost function $f_k : \mathcal{X} \to \mathbb{R}$ (assumed to be convex), and the online algorithm incurs a cost $f_k(\boldsymbol{x}_k)$. Since the online method selects $\boldsymbol{x}_k$ before seeing the cost, its decision might be suboptimal. So, the method suffers *regret*, which measures departure from the best (in hindsight) decision:

$$R_K := \sum\nolimits_{k=1}^{K} f_k(\boldsymbol{x}_k) - \min_{\boldsymbol{x}\in\mathcal{X}} \sum\nolimits_{k=1}^{K} f_k(\boldsymbol{x}). \tag{52}$$

In online optimization, we are interested in obtaining algorithms for which the average regret R_K/K is sublinear, so that as the number of rounds $K \to \infty$, the algorithm converges to the optimal answer. When the functions f_k are Lipschitz continuous but do not have any additional structure, regret $O(\sqrt{k})$ is the best attainable, and can be attained for example by the regularized follow-the-leader method [Hazan, 2011]. Assuming more structure, such as strong-convexity, logarithmic regret $O(\log k)$ can be attained—recent work of [Hazan and Kale, 2011] indicates that this might be the best possible.

A strong resemblance to the online setup is borne by *incremental algorithms*. The key difference here is that the components of the objective function are *fixed in advance*. Here too, the objective function is decomposable and we solve

$$\min_{\boldsymbol{x}\in\mathcal{X}} \quad \phi(\boldsymbol{x}) := \sum\nolimits_{i=1}^{m} f_k(\boldsymbol{x}), \tag{53}$$

where each $f_k : \mathcal{X} \to \mathbb{R}$ is convex. If we apply the projected (sub)gradient method to (53) we obtain the iteration

$$\boldsymbol{x}_{k+1} = P_{\mathcal{X}}\big(\boldsymbol{x}_k - \alpha_k \sum\nolimits_{i=1}^{m} f_k'(\boldsymbol{x}_k)\big), \quad f_k'(\boldsymbol{x}) \in \partial f_k(\boldsymbol{x}). \tag{54}$$

However, if the number of components m in (53) is very large, it can be impractical to compute the entire (sub)gradient at each iteration. Instead, we could use *incremental (sub)gradients* and simply iterate as

$$\boldsymbol{x}_{k+1} = P_{\mathcal{X}}\big(\boldsymbol{x}_k - \alpha_k f_{i(k)}'(\boldsymbol{x}_k)\big), \tag{55}$$

where $i(k) \in [1..m]$ is a suitably chosen index. (If $i(k)$ is picked uniformly at random, then (55) is essentially a stochastic subgradient method.) Numerical experience suggests that these methods can be much faster when one is far away from convergence, but become increasingly inferior close to convergence [Bertsekas, 2010]. These methods can also be shown to require $O(1/\epsilon^2)$ cycles (through all the m components) to converge to an ϵ-accuracy solution, though the constants in the big-Oh are much worse for a deterministic choice of $i(k)$ than for a randomized one.

6 Parallel and Distributed Optimization

Under preparation.

7 Algorithmic tricks of the trade

Under preparation.

8 A brief guide to software

Under preparation.

9 Summary

In this chapter we surveyed some fundamental algorithms and complexity results in large-scale convex optimization. Given the great importance of this class of optimization problems, especially for applications in data intensive areas such as machine learning, we could not cover all works that deserve a mention. In the notes we offer pointers to additional interesting material; we apologize in advance to the authors whose work has escaped our summary and will be grateful if they or the reader would alert us to such an omission.

9.1 Notes

This section is under constant revision...

Complexity. The complexity bounds discussed in this chapter draw on the seminal work of Nemirovsky and Yudin [1983]. However, we followed Nesterov's (2004) gentler exposition for greater accessibility. The lower bounds proved by [Nemirovsky and Yudin, 1983] invoke 'resisting oracles' (adversarial), which can require very ingenious construction. More recently, Raginsky and Rakhlin [2011] presented a simpler technique based on relating optimization to statistical hypothesis testing, which also yields several of the lower bounds for convex optimization.

Despite the acceleration achieved by the methods that exploiting problem structure to circumvent the black-box assumption, these lower-bounds and subgradient methods remain important as ever. This is so, because often the problem structure is too complex or too non-transparent to be exploited; and in such circumstances, subgradient schemes can still be employed—see [Nesterov, 2009] for more discussion.

Smooth convex optimization. As mentioned in Theorem 3.6, for sufficiently large problem dimension, the number of iterations of any first-order method cannot be better than $O(\sqrt{LD^2/\epsilon})$. This result was first shown by Nemirovsky and Yudin [1983], who also provided a nearly optimal method that achieved this lower-bound up to logarithmic factors. Nesterov's breakthrough work [Nesterov, 1983] provided the first provably optimal method. Later, Nesterov [1988, 2005a] presented two more optimal methods. Other variations on Nesterov's optimal methods have also been considered; for a nice summary and simplified explanation of such methods, we refer the reader to Tseng's (2008) manuscript.

Nonsmooth optimization. As hinted by Theorem 4.4, the number of iterations of any first-order method cannot be smaller than $O(L^2D^2/\epsilon^2)$. The simple subgradient method attains this bound (up to constant factors). Nemirovsky and Yudin [1983] presented the *mirror-descent* procedure that may be much more effective than ordinary subgradient method (which is merely mirror descent in a Euclidean setup) when dealing with non-Euclidean setups: the key idea is to use a different (non quadratic) 'prox-function' based on Bregman divergences. Nesterov [2005b] also allows using Bregman divergence based prox-functions.

In general, mirror-descent has been shown to be a workhorse in fact for optimally tackling all convex optimization settings, including strongly convex, smooth, stochastic, and online; moreover, its analysis can be simpler than that of Nesterov's optimal methods—we refer the reader to [Juditsky and Nemirovski, 2011a,b].

Stochastic optimization. Here the optimization problem may be written as one of minimizing an expected loss $\phi(\boldsymbol{x}) := \mathbb{E}[\Phi(\boldsymbol{x}, \xi)]$. The random variable ξ models parameter variation and uncertainty, and its value is not known; only its distribution is assumed to be available. The goal of stochastic programming is to find an $\boldsymbol{x}$ that minimizes ϕ on average, or with high probability. The expectation involves a multidimensional integral, so closed-form expressions for ϕ are almost never available. Instead, one has to resort to Monte-Carlo sampling, leading to the *sample average approximation* (SAA) and *stochastic approximation* (SA) methods. The SA method dates back to the classic paper of Robbins and Monro [1951]. Since then stochastic methods have exploded in interest; see [Nemirovsky and Yudin, 1983], [Kushner and Yin, 2003], [Nemirovski et al., 2009], [Ermoliev, 1981] and references therein.

It was believed that the SAA approach is superior to SA, but Nemirovski et al. [2009] derived a mirror-descent version of SA that exhibits an unimprovable rate of convergence $O((M+\sigma)/\sqrt{k})$; see also [Nesterov, 2009]. We remark in passing that in classic SA, the objective function ϕ is assumed to be smooth (C^2_L) and strongly convex, and only recently, interest in nonsmooth stochastic setups has surged—see also [Bertsekas and Tsitsiklis, 2000].

Algorithms. Figures 2 and 3 indicate that other first-order algorithms not discussed in this chapter can also be very competitive. Some noteworthy ones among these are: (i) alternating direction method of multipliers (ADMM)—see the recent survey treatment by [Boyd et al., 2011], who also advocate ADMM as a practical method for distributed and parallel optimization; (ii) coordinate

descent methods [Tseng and Yun, 2009, Nesterov, 2010]; (iii) trust-region methods for composite minimization [Kim et al., 2010]; (iv) interior point methods specialized to machine learning problems [Gondzio, 2011, Andersen et al., 2011] (though generally not scalable, unless the Hessian has special structure or is very sparse); (v) bundle and cutting plane methods [Teo et al., 2010]; (vi) specialized primal-dual methods [Esser et al., 2010]; (vii) level-set methods for composite optimization [Lan, 2011]; (viii) dual augmented Lagrangian methods [Tomioka et al., 2011]; (ix) Nesterov style methods with optimal rates despite inexact proximity operators [Schmidt et al., 2011].

Other setups. We mention some important setups missing in this chapter: (i) bandit optimization [Audibert and Munos, 2011]; (ii) nonconvex stochastic optimization [Bertsekas and Tsitsiklis, 2000, Ermoliev, 1981]; (iii) variational inequalities [Nemirovski, 2004]; (iv) robust optimization [Ben-Tal et al., 2009]; (v) derivative free minimization [Conn et al., 2009]; (vi) trust-region methods [Conn et al., 2000]; (vii) distributed and parallel optimization [Bertsekas, 2010, Zinkevich et al., 2010, Duchi et al., 2011, Recht et al., 2011].

Acknowledgments

I am grateful to Jeff Bilmes for hosting me at the University of Washington, Seattle, during my unforeseen visit in July 2011; large-parts of this chapter were written during that time.

References

M. S. Andersen, J. Dahl, Z. Liu, and L. Vandenberghe. Interior-point methods for large-scale cone programming. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for machine learning*. MIT Press, 2011.

J.-Y. Audibert and R. Munos. Introduction to Bandits: Algorithms and Theory. ICML 2011 Tutorial, 2011.

F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Convex optimization with sparsity-inducing norms. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*. MIT Press, 2011.

R. Baraniuk. Compressive sensing. *IEEE Signal Processing Magazine*, 24(4):118–121, 2007.

J. Barzilai and J. M. Borwein. Two-Point Step Size Gradient Methods. *IMA J. Num. Analy.*, 8(1), 1988.

A. Beck and M. Teboulle. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM J. Imaging Sciences*, 2(1):183–202, 2009.

A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton Univ. Press, 2009.

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

D. P. Bertsekas. Incremental Gradient, Subgradient, and Proximal Methods for Convex Optimization: A Survey. Technical Report LIDS-P-2848, MIT, 2010.

D. P. Bertsekas and J. N. Tsitsiklis. Gradient Convergence in Gradient methods with Errors. *SIAM J. on Optimization*, 10(3):627–642, 2000.

L. Bottou and O. Bousquet. The tradeoffs of large-scale learning. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for machine learning*. MIT Press, 2011.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. In Michael Jordan, editor, *Foundations and Trends in Machine Learning*, volume 3, pages 1–124. NOW, 2011.

J. Chen and S. Burer. A first-order smoothing technique for a class of large-scale linear programs. *Argonne National Labls Preprint*, (ANL/MCS-P1971-1011), 2011.

P. L. Combettes and J.-C. Pesquet. Proximal Splitting Methods in Signal Processing. *arXiv:0912.3522v4*, May 2010.

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

A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust-Region Methods*. SIAM, 2000.

A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to Derivative-Free Optimization*. SIAM, 2009.

J. Duchi, A. Agarwal, and M. Wainwright. Dual Averaging for Distributed Optimization: Convergence Analysis and Network Scaling . *IEEE Transactions on Automatic Control*, 2011. To appear.

Y. Ermoliev. Stochastic quasigradient methods and their applications in systems optimization. Technical Report WP-81-2, International Institute for Applied Systems Analysis, 1981.

E. Esser, X. Zhang, and T. F. Chan. A General Framework for a Class of First Order Primal-Dual Algorithms for Convex Optimization in Imaging Science. *SIAM J. Imaging Sciences*, 3(4): 1015–1046, 2010.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

S. Ghadimi and G. Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization. *SIAM J. Optimization*, 2010. Submitted.

J. Gondzio. Interior point methods in machine learning. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for machine learning*. MIT Press, 2011.

E. Hazan. The convex optimization approach to regret minimization. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for machine learning*. MIT Press, 2011.

E. Hazan and S. Kale. Beyond the regret minimization barrier: an optimal algorithm for stochastic strongly-convex optimization. In *Conference on Learning Theory (COLT)*, 2011.

A. Juditsky and A. Nemirovski. First-Order Methods for Nonsmooth Convex Large-Scale Optimization, I: General Purpose Methods. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for machine learning*. MIT Press, 2011a.

A. Juditsky and A. Nemirovski. First-Order Methods for Nonsmooth Convex Large-Scale Optimization, II: Utilizing Problem's Structure. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for machine learning*. MIT Press, 2011b.

D. Kim, S. Sra, and I. S. Dhillon. A scalable trust-region algorithm with application to mixed-norm regression. In *Int. Conf. Machine Learning (ICML)*, 2010. *Submitted*.

H. J. Kushner and G. G. Yin. *Stochastic approximation and recursive algorithms and applications*. Springer, 2003.

G. Lan. Level methods uniformly optimal for composite and structured nonsmooth convex optimization. *Submitted*, 2011.

J. Liu, S. Ji, and J. Ye. *SLEP: Sparse Learning with Efficient Projections*. Arizona State University, 2009.

A. Nemirovski. Prox-Method with Rate of Convergence $O(1/t)$ for Variational Inequalities with Lipschitz Continuous Monotone Operators and Smooth Convex-Concave Saddle Point Problems. *SIAM J. on Optimization*, 15:229–251, January 2004.

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust Stochastic Approximation Approach to Stochastic Programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

A. S. Nemirovsky and D. B. Yudin. *Problem complexity and method efficiency in optimization*. Wiley-Interscience, 1983. Translated by: E. R. Dawson.

Yu. Nesterov. A method for solving a convex programming problem with rate of convergence $o(1/k^2)$. *Soviet Math. Dokady*, 27(2):372–376, 1983.

Yu. Nesterov. On an approach to the construction of optimal methods of minimization of smooth convex functions. *Ekonom. i. Mat. Metody*, 24:509–517, 1988.

Yu. Nesterov. Smooth minimization of non-smooth functions. Technical Report 2003/12, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), January 2003.

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

Yu. Nesterov. Smooth minimization of nonsmooth functions. *Math. Program., Series A*, 103:127–152, 2005a.

Yu. Nesterov. Excessive Gap Technique in Nonsmooth Convex Minimization. *SIAM Journal on Optimization*, 16(1):235–249, 2005b.

Yu. Nesterov. Gradient methods for minimizing composite objective function. Technical Report 2007/76, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), September 2007.

Yu. Nesterov. How to advance in Structural Convex Optimization. *OPTIMA, MPS Newsletter*, 78: 2–5, 2008.

Yu. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120:221–259, 2009.

Yu. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. Technical Report 2010/2, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2010.

Yu. Nesterov and A. Nemirovski. *Interior Point Polynomial Algorithms in Convex Programming*. SIAM, 1994.

M. Patriksson. A survey on a classic core problem in operations research. Technical Report 2005:33, Chalmers University of Technology and Göteborg University, Oct. 2005.

B.T. Polyak. *Introduction to Optimization*. Optimization Software Inc., 1987.

Stephen Portnoy and Roger Koenker. The Gaussian hare and the Laplacian tortoise: computability of squared-error versus absolute-error estimators. *Statistical Science*, 12(4):279–300, 1997.

M. Raginsky and A. Rakhlin. Information-based complexity, feedback and dynamics in convex programming. *IEEE Transactions on Information Theory*, 2011. doi:10.1109/TIT.2011.2154375.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 693–701. 2011.

H. Robbins and S. Monro. A Stochastic Approximation Method. *Annals of Mathematical Statistics*, 22(3):400–407, 1951.

R. T. Rockafellar. *Convex Analysis*. Princeton Univ. Press, 1970.

R. T. Rockafellar and R. J.-B. Wets. *Variational analysis*. Springer, 1998.

Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

M. Schmidt, N. Le Roux, and F. Bach. Inexact. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

S. Sra, S. Nowozin, and S. J. Wright, editors. *Optimization for Machine Learning*. MIT Press, 2011.

N. Srebro and A. Tewari. Stochastic Optimization for Machine Learning. ICML 2010 Tutorial, 2010.

P. Stobbe and A. Krause. Efficient minimization of decomposable submodular functions. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

C. H. Teo, S.V.N. Vishwanthan, A. J. Smola, and Q. V. Le. Bundle Methods for Regularized Risk Minimization. *J. Machine Learning Research*, 11:311–365, 2010.

R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *J. Royal Stat. Soc.: Series B*, 67(1):91–108(18), 2005.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *J. R. Statist. Soc.*, 58(1): 267–288, 1996.

V. M. Tikhomirov. The evolution of methods of convex optimization. *The American Mathematical Monthly*, 103(1):pp. 65–71, 1996.

R. Tomioka, T. Suzuki, and M. Sugiyama. Super-Linear Convergence of Dual Augmented Lagrangian Algorithm for Sparsity Regularized Estimation. *J. Machine Learning Research*, 12:1537–1586, 2011.

P. Tseng. On accelerated gradient methods for convex-concave minimization. *SIAM J. Optimization*, 2008. Submitted.

P. Tseng and S. Yun. A Block-Coordinate Gradient Descent Method for Linearly Constrained Nonsmooth Separable Optimization. *J. Optim. Theory Appl.*, 140:513–535, 2009.

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

Yiming Ying, Kaizhu Huang, and Colin Campbell. Sparse Metric Learning via Smooth Optimization. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 2214–2222. 2009.

D. B. Yudin and A. S. Nemirovskii. Informational complexity and effective methods of solution for convex extremal problems. *Ekonomika i Matematicheski Metody*, 12:357–369, 1976.

X. Zhang, A. Saha, and S. V. N. Vishwanathan. Accelerated Training of Max-Margin Markov Networks with Kernels. In *Algorithmic Learning Theory (ALT)*, 2011.

M. Zinkevich, M. Weimer, A. Smola, and L. Li. Parallelized Stochastic Gradient Descent. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2595–2603. 2010.