Optimization methods for large-scale machine learning

Alberto Bietti and Julien Mairal

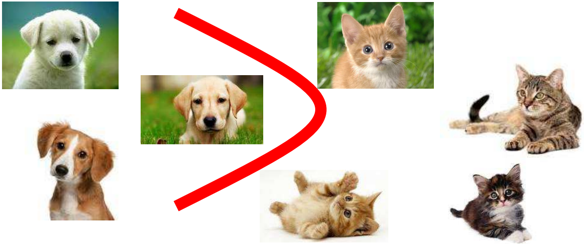
Inria Grenoble

Autrans, SMAI-MODE, 2018
Part II

Common paradigm: optimization for machine learning

Optimization is central to machine learning. For instance, in supervised learning, the goal is to learn a **prediction function** $f : \mathcal{X} \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\ldots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

$$\min_{f\in\mathcal{F}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda\Omega(f)}_{\text{regularization}} .$$



[Vapnik, 1995, Bottou, Curtis, and Nocedal, 2016]...

Focus of this part

Minimizing large finite sums

Consider the minimization of a large sum of convex functions

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(x) + \psi(x) \right\},$$

where each f_i is **L-smooth and convex** and ψ is a convex regularization penalty but not necessarily differentiable.

Focus of this part

Minimizing large finite sums

Consider the minimization of a large sum of convex functions

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(x) + \psi(x) \right\},$$

where each f_i is **L-smooth and convex** and ψ is a convex regularization penalty but not necessarily differentiable.

Why this setting?

- convexity makes it easy to obtain **complexity** bounds.
- convex optimization is often effective for non-convex problems.

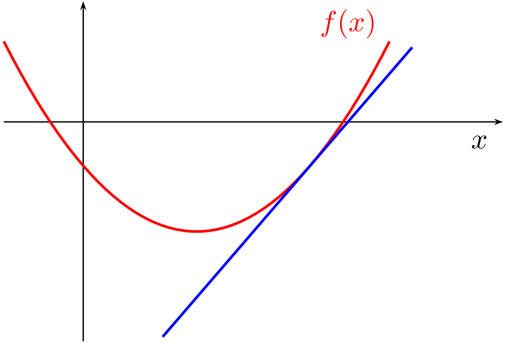
What we will not cover

- performance of approaches in terms of test error.

Introduction of a few optimization principles

Convex Functions

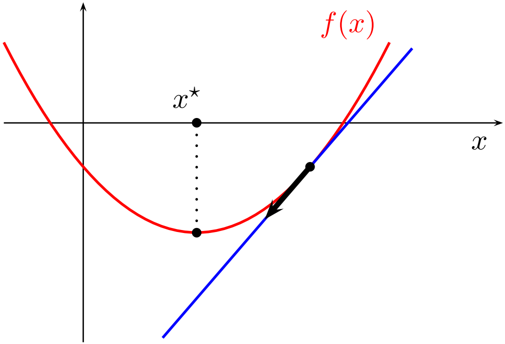
Why do we care about convexity?



Introduction of a few optimization principles

Convex Functions

Local observations give information about the global optimum

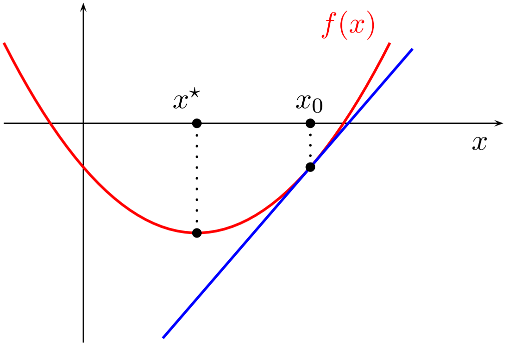


- $\nabla f(x) = 0$ is a necessary and sufficient optimality condition for differentiable convex functions;
- it is often easy to upper-bound $f(x) - f^\star$.

Introduction of a few optimization principles

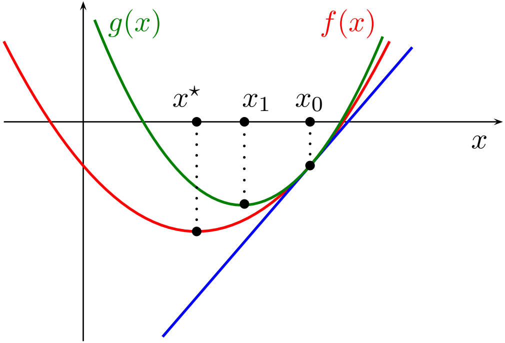
An important inequality for L-smooth convex functions

If f is convex and smooth



- $f(x) \geq \underbrace{f(x_0) + \nabla f(x_0)^\top (x - x_0)}_{\text{linear approximation}}$;
- if f is non-smooth, a similar inequality holds for subgradients.

If ∇f is L-Lipschitz continuous (f does not need to be convex)

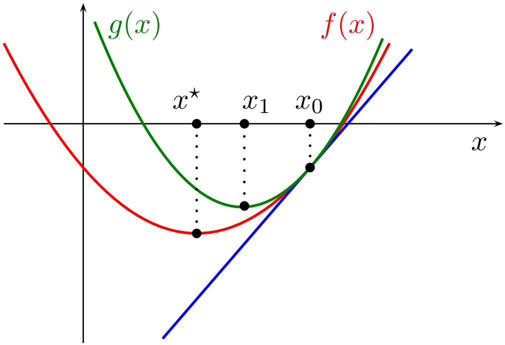


- $f(x) \leq g(x) = \underbrace{f(x_0) + \nabla f(x_0)^\top (x - x_0)}_{\text{linear approximation}} + \frac{L}{2}\|x - x_0\|_2^2$;

Introduction of a few optimization principles

An important inequality for smooth functions

If ∇f is L-Lipschitz continuous (f does not need to be convex)

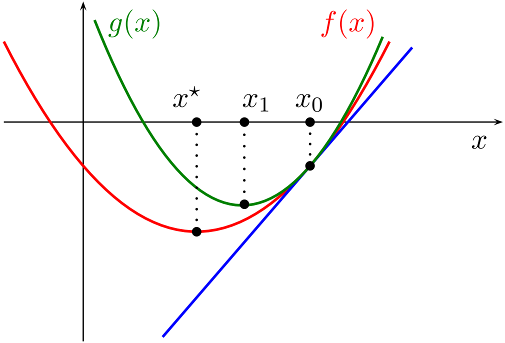


- $f(x) \leq g(x) = \underbrace{f(x_0) + \nabla f(x_0)^\top (x - x_0)}_{\text{linear approximation}} + \frac{L}{2}\|x - x_0\|_2^2$;
- $g(x) = C_{x_0} + \frac{L}{2}\|x_0 - (1/L)\nabla f(x_0) - x\|_2^2$.

Introduction of a few optimization principles

An important inequality for smooth functions

If ∇f is L-Lipschitz continuous (f does not need to be convex)



- $f(x) \leq g(x) = \underbrace{f(x_0) + \nabla f(x_0)^\top (x - x_0)}_{\text{linear approximation}} + \frac{L}{2}\|x - x_0\|_2^2$;
- $x_1 = x_0 - \frac{1}{L}\nabla f(x_0)$. (gradient descent step).

Introduction of a few optimization principles

Gradient Descent Algorithm

Assume that f is convex and L-smooth (∇f is L-Lipschitz).

Theorem

Consider the algorithm

$$\boxed{x_t \leftarrow x_{t-1} - \frac{1}{L}\nabla f(x_{t-1}).}$$

Then,

$$f(x_t) - f^\star \leq \frac{L\|x_0 - x^\star\|_2^2}{2t}.$$

Proof (1/2)

Proof of the main inequality for smooth functions

We want to show that for all x and z,

$$f(x) \leq f(z) + \nabla f(z)^\top (x - z) + \frac{L}{2}\|x - z\|_2^2.$$

Proof (1/2)

Proof of the main inequality for smooth functions

We want to show that for all x and z,

$$f(x) \leq f(z) + \nabla f(z)^\top (x - z) + \frac{L}{2}\|x - z\|_2^2.$$

By using Taylor's theorem with integral form,

$$f(x) - f(z) = \int_0^1 \nabla f(tx + (1-t)z)^\top (x - z)dt.$$

Then,

$$\begin{aligned}
f(x)-f(z)-\nabla f(z)^\top(x-z) &\leq \int_0^1 (\nabla f(tx+(1-t)z)-\nabla f(z))^\top (x-z)dt \\
&\leq \int_0^1 |(\nabla f(tx+(1-t)z)-\nabla f(z))^\top (x-z)|dt \\
&\leq \int_0^1 \|\nabla f(tx+(1-t)z)-\nabla f(z)\|_2 \|x-z\|_2 dt \quad \text{(C.-S.)} \\
&\leq \int_0^1 Lt\|x-z\|_2^2 dt = \frac{L}{2}\|x-z\|_2^2.
\end{aligned}$$

Proof (2/2)

Proof of the theorem

We have shown that for all x,

$$f(x) \leq g_t(x) = f(x_{t-1}) + \nabla f(x_{t-1})^\top (x - x_{t-1}) + \frac{L}{2}\|x - x_{t-1}\|_2^2.$$

g_t is minimized by x_t; it can be rewritten $g_t(x) = g_t(x_t) + \frac{L}{2}\|x - x_t\|_2^2$. Then,

$$\begin{aligned} f(x_t) \leq g_t(x_t) &= g_t(x^\star) - \frac{L}{2}\|x^\star - x_t\|_2^2 \\ &= f(x_{t-1}) + \nabla f(x_{t-1})^\top (x^\star - x_{t-1}) + \frac{L}{2}\|x^\star - x_{t-1}\|_2^2 - \frac{L}{2}\|x^\star - x_t\|_2^2 \\ &\leq f^\star + \frac{L}{2}\|x^\star - x_{t-1}\|_2^2 - \frac{L}{2}\|x^\star - x_t\|_2^2. \end{aligned}$$

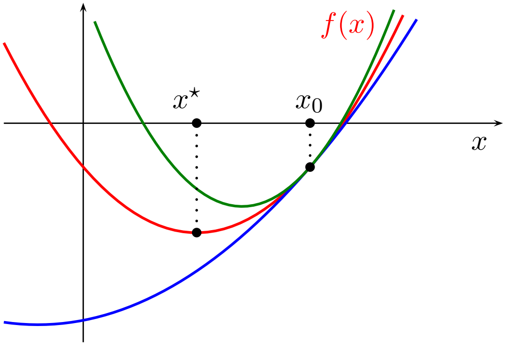
By summing from $t = 1$ to T, we have a telescopic sum

$$T(f(x_T) - f^\star) \leq \sum_{t=1}^{T} f(x_t) - f^\star \leq \frac{L}{2}\|x^\star - x^0\|_2^2 - \frac{L}{2}\|x^\star - x_T\|_2^2.$$

Introduction of a few optimization principles

An important inequality for smooth and μ-strongly convex functions

If ∇f is L-Lipschitz continuous and f μ-strongly convex



- $f(x) \leq f(x_0) + \nabla f(x_0)^\top (x - x_0) + \frac{L}{2}\|x - x_0\|_2^2$;
- $f(x) \geq f(x_0) + \nabla f(x_0)^\top (x - x_0) + \frac{\mu}{2}\|x - x_0\|_2^2$;

Introduction of a few optimization principles

Proposition

When f is μ-strongly convex and L-smooth, the gradient descent algorithm with step-size $1/L$ produces iterates such that

$$f(x_t) - f^\star \leq \left(1 - \frac{\mu}{L}\right)^t \frac{L\|x_0 - x^\star\|_2^2}{2}.$$

We call that a **linear** convergence rate.

Remarks

- if f is twice differentiable, L and μ represent the larget and smallest eigenvalues of the Hessian, respectively.
- L/μ is called the **condition number**.

Proof

We start from an inequality from the previous proof

$$
\begin{aligned}
f(x_t) &\leq f(x_{t-1}) + \nabla f(x_{t-1})^\top (x^\star - x_{t-1}) + \frac{L}{2}\|x^\star - x_{t-1}\|_2^2 - \frac{L}{2}\|x^\star - x_t\|_2^2 \\
&\leq f^\star + \frac{L-\mu}{2}\|x^\star - x_{t-1}\|_2^2 - \frac{L}{2}\|x^\star - x_t\|_2^2.
\end{aligned}
$$

In addition, we have that $f(x_t) \geq f^\star + \frac{\mu}{2}\|x_t - x^\star\|_2^2$, and thus

$$
\begin{aligned}
\|x^\star - x_t\|_2^2 &\leq \frac{L-\mu}{L+\mu}\|x^\star - x_{t-1}\|_2^2 \\
&\leq \left(1 - \frac{\mu}{L}\right)\|x^\star - x_{t-1}\|_2^2.
\end{aligned}
$$

Finally,

$$
\begin{aligned}
f(x_t) - f^\star &\leq \frac{L}{2}\|x_t - x^\star\|_2^2 \\
&\leq \left(1 - \frac{\mu}{L}\right)^t \frac{L\|x^\star - x_0\|_2^2}{2}
\end{aligned}
$$

Introduction of a few optimization principles

Remark: with stepsize $1/L$, gradient descent may be interpreted as a **majorization-minimization** algorithm:

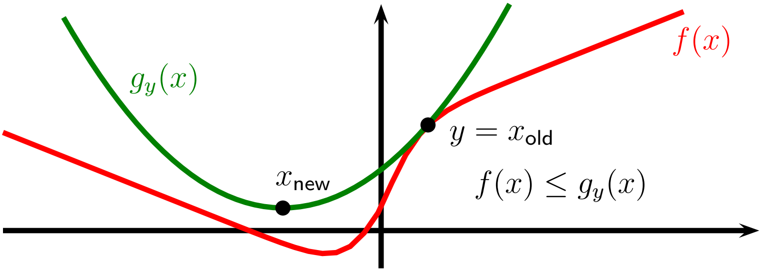
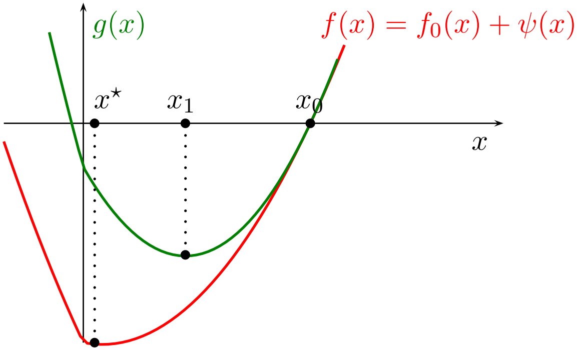


Figure: At each step, we update $x \in \arg\min_{x \in \mathbb{R}^p} g_y(x)$

The proximal gradient method

An important inequality for composite functions

If ∇f_0 is L-Lipschitz continuous



- $f_0(x) + \psi(x) \leq f_0(x_0) + \nabla f_0(x_0)^\top (x - x_0) + \frac{L}{2}\|x - x_0\|_2^2 + \psi(x)$;
- x_1 minimizes g.

The proximal gradient method

Gradient descent for minimizing f consists of

$$x_t \leftarrow \arg\min_{x \in \mathbb{R}^p} g_t(x) \quad \Longleftrightarrow \quad x_t \leftarrow x_{t-1} - \frac{1}{L}\nabla f(x_{t-1}).$$

The proximal gradient method for minimizing $f = f_0 + \psi$ consists of

$$x_t \leftarrow \arg\min_{x \in \mathbb{R}^p} g_t(x),$$

which is equivalent to

$$\boxed{x_t \leftarrow \arg\min_{x \in \mathbb{R}^p} \frac{1}{2}\left\| x_{t-1} - \frac{1}{L}\nabla f_0(x_{t-1}) - x \right\|_2^2 + \frac{1}{L}\psi(x).}$$

It requires computing efficiently the **proximal operator** of ψ.

$$y \mapsto \arg\min_{x \in \mathbb{R}^p} \; \frac{1}{2}\|y - x\|_2^2 + \psi(x).$$

The proximal gradient method

Remarks

- also known as **forward-backward** algorithm;
- has similar convergence rates as the gradient descent method (the proof is nearly identical).
- there exists **line search schemes** to automatically tune L;

The case of ℓ_1

The proximal operator of $\lambda\|.\|_1$ is the soft-thresholding operator

$$x[j] = \text{sign}(y[j])(|y[j]| - \lambda)^+.$$

The resulting algorithm is called **iterative soft-thresholding**.

[Nowak and Figueiredo, 2001, Daubechies et al., 2004, Combettes and Wajs, 2006, Beck and Teboulle, 2009, Wright et al., 2009, Nesterov, 2013]...

The proximal gradient method

The proximal operator for the group Lasso penalty

$$\min_{x \in \mathbb{R}^p} \frac{1}{2}\|y - x\|_2^2 + \lambda \sum_{g \in \mathcal{G}} \|x[g]\|_q.$$

For $q = 2$,

$$x[g] = \frac{y[g]}{\|y[g]\|_2}(\|y[g]\|_2 - \lambda)^+, \quad \forall g \in \mathcal{G}.$$

For $q = \infty$,

$$x[g] = y[g] - \Pi_{\|.\|_1 \leq \lambda}[y[g]], \quad \forall g \in \mathcal{G}.$$

These formula generalize soft-thresholding to groups of variables.

The proximal gradient method

A few proximal operators:

- ℓ_0-penalty: hard-thresholding;
- ℓ_1-norm: soft-thresholding;
- group-Lasso: group soft-thresholding;
- fused-lasso (1D total variation): [Hoefling, 2010];
- total variation: [Chambolle and Darbon, 2009];
- hierarchical norms: [Jenatton et al., 2011], $O(p)$ complexity;
- overlapping group Lasso with ℓ_∞-norm: [Mairal et al., 2010];

Accelerated gradient descent methods

Nesterov introduced in the 80's an acceleration scheme for the gradient descent algorithm. It was generalized later to the composite setting.

FISTA

$$x_t \leftarrow \arg\min_{x \in \mathbb{R}^p} \frac{1}{2} \left\| x - \left(y_{t-1} - \frac{1}{L} \nabla f_0(y_{t-1}) \right) \right\|_2^2 + \frac{1}{L}\psi(x);$$

$$\text{Find } \alpha_t > 0 \;\; \text{s.t.} \;\; \alpha_t^2 = (1-\alpha_t)\alpha_{t-1}^2 + \frac{\mu}{L}\alpha_t;$$

$$y_t \leftarrow x_t + \beta_t(x_t - x_{t-1}) \quad \text{with} \quad \beta_t = \frac{\alpha_{t-1}(1-\alpha_{t-1})}{\alpha_{t-1}^2 + \alpha_t}.$$

- $f(x_t) - f^\star = O(1/t^2)$ for **convex** problems;
- $f(x_t) - f^\star = O((1 - \sqrt{\mu/L})^t)$ for μ-**strongly convex** problems;
- Acceleration works in many practical cases.

see [Beck and Teboulle, 2009, Nesterov, 1983, 2004, 2013]

What do we mean by "acceleration"?

Complexity analysis for large finite sums

Since f is a sum of n functions, computing ∇f requires computing n gradients ∇f_i. The complexity to reach an $\varepsilon-$solution is given below

	$\mu > 0$	$\mu = 0$
ISTA	$O\left(n\frac{L}{\mu}\log\left(\frac{1}{\varepsilon}\right)\right)$	$O\left(\frac{nL}{\varepsilon}\right)$
FISTA	$O\left(n\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$	$O\left(n\sqrt{\frac{L}{\varepsilon}}\right)$

Remarks

- ε-solution means here $f(x_t) - f^\star \leq \varepsilon$.
- For $n = 1$, the rates of FISTA are optimal for a "first-order local black box" [Nesterov, 2004].
- For $L = 1$ and $\mu = 1/n$, scales at best in $n^{3/2}$.

How does "acceleration" work?

Unfortunately, the literature does not provide any simple geometric explanation...

How does “acceleration” work?

Unfortunately, the literature does not provide any simple geometric explanation... but they are a few obvious facts and a mechanism introduced by Nesterov, called **“estimate sequence”**.

Obvious fact

- Simple gradient descent steps are “blind” to the past iterates, and are based on a **purely local** model of the objective.
- Accelerated methods usually involve an **extrapolation step** $y_t = x_t + \beta_t(x_t - x_{t-1})$ with β_t in $(0,1)$.
- Nesterov interprets acceleration as relying on a better model of the objective called **estimate sequence**.

How does "acceleration" work?

Definition of estimate sequence [Nesterov].

A pair of sequences $(\varphi_t)_{t\geq 0}$ and $(\lambda_t)_{t\geq 0}$, with $\lambda_t \geq 0$ and $\varphi_t : \mathbb{R}^p \to \mathbb{R}$, is called an **estimate sequence** of function f if $\lambda_t \to 0$ and

$$\text{for any } x \in \mathbb{R}^p \text{ and all } t \geq 0, \quad \varphi_t(x) - f(x) \leq \lambda_t(\varphi_0(x) - f(x)).$$

In addition, if for some sequence $(x_t)_{t\geq 0}$ we have

$$f(x_t) \leq \varphi_t^\star \triangleq \min_{x\in\mathbb{R}^p} \varphi_t(x),$$

then

$$f(x_t) - f^\star \leq \lambda_t(\varphi_0(x^\star) - f^\star),$$

where $x^\star$ is a minimizer of f.

How does “acceleration” work?

In summary, we need two properties

1. $\varphi_t(x) \leq (1 - \lambda_t) f(x) + \lambda_t \varphi_0(x)$;
2. $f(x_t) \leq \varphi_t^\star \triangleq \min_{x \in \mathbb{R}^p} \varphi_t(x)$.

Remarks

- φ_t is neither an upper-bound, nor a lower-bound;
- Finding the right estimate sequence is often nontrivial.

How does "acceleration" work?

In summary, we need two properties

1. $\varphi_t(x) \leq (1-\lambda_t) f(x) + \lambda_t \varphi_0(x)$;
2. $f(x_t) \leq \varphi_t^\star \triangleq \min_{x \in \mathbb{R}^p} \varphi_t(x)$.

How to build an estimate sequence?

Define φ_t recursively

$$\varphi_t(x) \triangleq (1-\alpha_t)\varphi_{t-1}(x) + \alpha_t d_t(x),$$

where d_t is a **lower-bound**, e.g., if f is smooth,

$$d_t(x) \triangleq f(y_t) + \nabla f(y_t)^\top (x - y_t) + \frac{\mu}{2}\|x - y_t\|_2^2,$$

Then, work hard to choose α_t as large as possible, and y_t and x_t such that property 2 holds. Subsequently, $\lambda_t = \prod_{t=1}^{t}(1-\alpha_t)$.

The stochastic (sub)gradient descent algorithm

Consider now the minimization of an expectation

$$\min_{x \in \mathbb{R}^p} f(x) = \mathbb{E}_z[\ell(x, z)],$$

To simplify, we assume that for all z, $x \mapsto \ell(x, z)$ is differentiable.

Algorithm

At iteration t,

- Randomly draw one example z_t from the training set;
- Update the current iterate

$$x_t \leftarrow x_{t-1} - \eta_t \nabla f_t(x_{t-1}) \quad \text{with} \quad f_t(x) = \ell(x, z_t).$$

- Perform online averaging of the iterates (optional)

$$\tilde{x}_t \leftarrow (1 - \gamma_t)\tilde{x}_{t-1} + \gamma_t x_t.$$

The stochastic (sub)gradient descent algorithm

There are various learning rates strategies (constant, varying step-sizes), and averaging strategies. Depending on the problem assumptions and choice of η_t, γ_t, classical convergence rates may be obtained:

- $f(\tilde{x}_t) - f^\star = O(1/\sqrt{t})$ for convex problems;
- $f(\tilde{x}_t) - f^\star = O(1/t)$ for strongly-convex ones;

Remarks

- The convergence rates are not great, but the complexity **per-iteration** is small (1 gradient evaluation for minimizing an empirical risk versus n for the batch algorithm).
- When the amount of data is infinite, the method **minimizes the expected risk** (which is what we want).
- Choosing a good learning rate automatically is an open problem.

Proof of an $O(1/\sqrt{t})$ rate for the convex case

Inspired by (aka, stolen from) F. Bach's slides

Assumptions

- The solution lies in a bounded domain $\mathcal{C} = \{\|x\| \leq D\}$.
- The sub-gradients are bounded on $\mathcal{C}$: $\|\nabla f_t(x)\| \leq B$.
- Fix in advance the number of iterations T and choose $\eta_t = \frac{2D}{B\sqrt{T}}$.
- Choose Polyak-Ruppert averaging $\tilde{x}_T = (1/T)\sum_{t=0}^{T-1} x_t$.
- Perform updates with projections

$$x_t \leftarrow \Pi_{\mathcal{C}}[x_{t-1} - \eta_t \nabla f_t(x_{t-1})].$$

Proposition

$$\mathbb{E}[f(\tilde{x}_t) - f^\star] \leq \frac{2DB}{\sqrt{T}}.$$

Proof of an $O(1/\sqrt{t})$ rate for the convex case

Inspired by (aka, stolen from) F. Bach's slides

- $\mathcal{F}_t$: information up to time t.
- $\|x\| \leq D$ and $\|\nabla f_t(x)\| \leq B$. Besides $\mathbb{E}[\nabla f_t(x)|\mathcal{F}_{t-1}] = \nabla f(x)$.

$$\begin{aligned}\|x_t - x^\star\|^2 &\leq \|x_{t-1} - \eta_t \nabla f_t(x_{t-1}) - x^\star\|^2 \\ &\leq \|x_{t-1} - x^\star\|^2 + B^2\eta_t^2 - 2\eta_t(x_{t-1} - x^\star)^\top \nabla f_t(x_{t-1}).\end{aligned}$$

Take now **conditional expectations**

$$\begin{aligned}\mathbb{E}[\|x_t - x^\star\|^2|\mathcal{F}_{t-1}] &\leq \|x_{t-1} - x^\star\|^2 + B^2\eta_t^2 - 2\eta_t(x_{t-1} - x^\star)^\top \nabla f(x_{t-1}) \\ &\leq \|x_{t-1} - x^\star\|^2 + B^2\eta_t^2 - 2\eta_t(f(x_{t-1}) - f^\star).\end{aligned}$$

Take now **full expectations**

$$\mathbb{E}[\|x_t - x^\star\|^2] \leq \mathbb{E}[\|x_{t-1} - x^\star\|^2] + B^2\eta_t^2 - 2\eta_t\mathbb{E}[f(x_{t-1}) - f^\star],$$

and, after reorganizing the terms

$$\mathbb{E}[f(x_{t-1}) - f^\star] \leq \frac{B^2\eta_t^2}{2} + \frac{1}{2\eta_t}\left(\mathbb{E}[\|x_{t-1} - x^\star\|^2] - \mathbb{E}[\|x_t - x^\star\|^2]\right).$$

Proof of an $O(1/\sqrt{t})$ rate for the convex case

Inspired by (aka, stolen from) F. Bach's slides

We start again from

$$\mathbb{E}[f(x_{t-1}) - f^\star] \leq \frac{B^2\eta_t^2}{2} + \frac{1}{2\eta_t}\left(\mathbb{E}[\|x_{t-1} - x^\star\|^2] - \mathbb{E}[\|x_t - x^\star\|^2]\right).$$

and we exploit the telescopic sum

$$\sum_{t=1}^{T}\mathbb{E}[f(x_{t-1}) - f^\star] \leq \sum_{t=1}^{T}\frac{B^2\eta_t^2}{2} + \sum_{t=1}^{T}\frac{1}{2\eta_t}\left(\mathbb{E}[\|x_{t-1} - x^\star\|^2] - \mathbb{E}[\|x_t - x^\star\|^2]\right)$$

$$\leq T\frac{B^2\eta^2}{2} + \frac{4D^2}{2\eta} \leq 2DB\sqrt{T} \quad \text{with} \quad \gamma = \frac{2D}{B\sqrt{T}}.$$

Finally, we conclude by using a convexity inequality

$$\mathbb{E}f\left(\frac{1}{T}\sum_{t=0}^{T-1}\right) - f^\star \leq \frac{2DB}{\sqrt{T}}.$$

Constant step-size SGD for the strongly convex case

- Gradient "variance": $\mathbb{E}[\|\nabla f_t(x)\|^2] \leq B^2$

$$\|x_t - x^*\|^2 = \|x_{t-1} - x^*\|^2 - 2\eta_t \nabla f_t(x_{t-1})^\top (x_{t-1} - x^*) + \eta_t^2 \|\nabla f_t(x_{t-1})\|^2$$

$$\mathbb{E}[\|x_t - x^*\|^2 | \mathcal{F}_{t-1}] = \|x_{t-1} - x^*\|^2 - 2\eta_t \nabla f(x_{t-1})^\top (x_{t-1} - x^*) + \eta_t^2 B^2$$
$$\leq (1 - \mu\eta_t)\|x_{t-1} - x^*\|^2 - 2\eta_t (f(x_{t-1}) - f^*) + \eta_t^2 B^2$$

- **Constant step-size** η, no averaging:

$$\mathbb{E}[\|x_t - x^*\|^2] \leq (1 - \mu\eta)\mathbb{E}[\|x_{t-1} - x^*\|^2] + \eta^2 B^2$$
$$\xrightarrow[t\to\infty]{} \frac{\eta B^2}{\mu} \quad \text{(with linear rate)}$$

- Can replace B^2 with variance σ^2 for smooth f if $\eta \leq 1/L$
- Limit value becomes smaller with:
  - Smaller step-size: $\eta \to \eta/m$ (**but** m times slower rate)
  - Mini-batch of size m: $\sigma^2 \to \sigma^2/m$ (**but** m times more computation)

$O(1/t)$ for the strongly convex case

- From the previous slide:

$$\mathbb{E}[\|x_t - x^*\|^2] \leq (1 - \mu\eta_t)\mathbb{E}[\|x_{t-1} - x^*\|^2] + \eta_t^2 B^2$$

- Take $\eta_t = \frac{2}{\mu(\gamma+t)}$ (with $\eta_1 \leq 1/L$) and by induction:

$$\mathbb{E}[\|x_t - x^*\|^2] \leq \frac{\nu}{\gamma + t + 1}, \quad \nu := \max\left\{\frac{4B^2}{\mu^2}, (\gamma+1)\|x_0 - x^*\|^2\right\}$$

- $f(x_t) - f(x^*) \leq \frac{L}{2}\|x_t - x^*\|^2$
- Start with constant step-size to forget initial condition faster
- Averaging improves from $O(LB^2/\mu^2 t)$ to $O(B^2/\mu t)$

Back to finite sums

Consider now the case of interest for us today:

$$\min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} f_i(x),$$

Question

Can we do as well as SGD in terms of cost per iteration, while enjoying a fast (linear) convergence rate like (accelerated or not) gradient descent?

For $n = 1$, no!

The rates are optimal for a "first-order local black box" [Nesterov, 2004].

For $n \geq 1$, yes! We need to design algorithms

- whose per-iteration **computational complexity** is smaller than n;
- whose **convergence rate** may be worse than FISTA....
- ...but with a better expected **computational complexity**.

Incremental gradient descent methods

$$\min_{x \in \mathbb{R}^p} \left\{ f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}.$$

Several **randomized** algorithms are designed with one ∇f_i computed per iteration, with **fast convergence rates**, e.g., SAG [Schmidt et al., 2013]:

$$x_k \leftarrow x_{k-1} - \frac{\gamma}{Ln} \sum_{i=1}^{n} y_i^k \quad \text{with} \quad y_i^k = \begin{cases} \nabla f_i(x_{k-1}) & \text{if } \; i = i_k \\ y_i^{k-1} & \text{otherwise} \end{cases}.$$

Incremental gradient descent methods

$$\min_{x \in \mathbb{R}^p} \left\{ f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}.$$

Several **randomized** algorithms are designed with one ∇f_i computed per iteration, with **fast convergence rates**, e.g., SAG [Schmidt et al., 2013]:

$$x_k \leftarrow x_{k-1} - \frac{\gamma}{Ln} \sum_{i=1}^{n} y_i^k \quad \text{with} \quad y_i^k = \begin{cases} \nabla f_i(x_{k-1}) & \text{if} \;\; i = i_k \\ y_i^{k-1} & \text{otherwise} \end{cases}.$$

See also SVRG, SAGA, SDCA, MISO, Finito...
Some of these algorithms perform updates of the form

$$x_k \leftarrow x_{k-1} - \eta_k g_k \quad \text{with} \quad \mathbb{E}[g_k] = \nabla f(x_{k-1}),$$

but g_k has **lower variance** than in SGD.

[Schmidt et al., 2013, Xiao and Zhang, 2014, Defazio et al., 2014a,b, Shalev-Shwartz and Zhang, 2012, Mairal, 2015, Zhang and Xiao, 2015]

Incremental gradient descent methods

These methods achieve low **(worst-case)** complexity in expectation.
The number of gradients evaluations to ensure $f(x_k) - f^\star \leq \varepsilon$ is

	$\mu > 0$
FISTA	$O\left(n\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$
SVRG, SAG, SAGA, SDCA, MISO, Finito	$O\left(\max\left(n, \frac{\bar{L}}{\mu}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$

Incremental gradient descent methods

These methods achieve low **(worst-case)** complexity in expectation.
The number of gradients evaluations to ensure $f(x_k) - f^\star \leq \varepsilon$ is

	$\mu > 0$
FISTA	$O\left(n\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$
SVRG, SAG, SAGA, SDCA, MISO, Finito	$O\left(\max\left(n, \frac{\bar{L}}{\mu}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$

Main features vs. stochastic gradient descent

- Same complexity per-iteration (but higher memory footprint).
- **Faster convergence** (exploit the finite-sum structure).
- **Less parameter tuning** than SGD.
- Some variants are **compatible with a composite term** ψ.

Incremental gradient descent methods

These methods achieve low **(worst-case)** complexity in expectation.
The number of gradients evaluations to ensure $f(x_k) - f^\star \leq \varepsilon$ is

	$\mu > 0$
FISTA	$O\left(n\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$
SVRG, SAG, SAGA, SDCA, MISO, Finito	$O\left(\max\left(n, \frac{\bar{L}}{\mu}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$

Important remarks

- When $f_i(x) = \ell(z_i^\top x)$, the memory footprint is $O(n)$ otherwise $O(dn)$, except for SVRG ($O(d)$).
- Some algorithms require an estimate of μ;
- $\bar{L}$ is the average (or max) of the Lipschitz constants of the ∇f_i's.
- The L for fista is the Lipschitz constant of ∇f: $L \leq \bar{L}$.

Incremental gradient descent methods

stealing again a bit from F. Bach's slides.

Variance reduction

Consider two random variables X, Y and define

$$Z = X - Y + \mathbb{E}[Y].$$

Then,

- $\mathbb{E}[Z] = \mathbb{E}[X]$
- $\text{Var}(Z) = \text{Var}(X) + \text{Var}(Y) - 2\text{cov}(X, Y)$.

The variance of Z may be smaller if X and Y are positively correlated.

Incremental gradient descent methods

stealing again a bit from F. Bach's slides.

Variance reduction

Consider two random variables X, Y and define

$$Z = X - Y + \mathbb{E}[Y].$$

Then,

- $\mathbb{E}[Z] = \mathbb{E}[X]$
- $\text{Var}(Z) = \text{Var}(X) + \text{Var}(Y) - 2\text{cov}(X, Y)$.

The variance of Z may be smaller if X and Y are positively correlated.

Why is it useful for stochastic optimization?

- step-sizes for SGD have to decrease to ensure convergence.
- with variance reduction, one may use constant step-sizes.

Incremental gradient descent methods

SVRG

$$x_t = x_{t-1} - \gamma \left(\nabla f_{i_t}(x_{t-1}) - \nabla f_{i_t}(y) + \nabla f(y) \right),$$

where y is updated every epoch and $\mathbb{E}[\nabla f_{i_t}(y) | \mathcal{F}_{t-1}] = \nabla f(y)$.

SAGA

$$x_t = x_{t-1} - \gamma \left(\nabla f_{i_t}(x_{t-1}) - y_{i_t}^{t-1} + \frac{1}{n} \sum_{i=1}^n y_i^{t-1} \right),$$

where $\mathbb{E}[y_{i_t}^{t-1} | \mathcal{F}_{t-1}] = \frac{1}{n} \sum_{i=1}^n y_i^{t-1}$ and $y_i^t = \begin{cases} \nabla f_i(x_{t-1}) & \text{if } i = i_t \\ y_i^{t-1} & \text{otherwise.} \end{cases}$

MISO/Finito: for $n \geq L/\mu$, same form as SAGA but

$$\frac{1}{n} \sum_{i=1}^n y_i^{t-1} = -\mu x_{t-1} \quad \text{and} \quad y_i^t = \begin{cases} \nabla f_i(x_{t-1}) - \mu x_{t-1} & \text{if } i = i_t \\ y_i^{t-1} & \text{otherwise.} \end{cases}$$

Can we do even better for large finite sums?

Without vs with acceleration

	$\mu > 0$
FISTA	$O\left(n\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$
SVRG, SAG, SAGA, SDCA, MISO, Finito	$O\left(\max\left(n, \frac{\bar{L}}{\mu}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$
Accelerated versions	$\tilde{O}\left(\max\left(n, \sqrt{n\frac{\bar{L}}{\mu}}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$

- Acceleration for specific algorithms [Shalev-Shwartz and Zhang, 2014, Lan, 2015, Allen-Zhu, 2016].
- Generic acceleration: Catalyst [Lin et al., 2015].
- see [Agarwal and Bottou, 2015] for discussions about optimality.

What we have not (or should have) covered

Import approaches and concepts

- distributed optimization.
- proximal splitting / ADMM.
- Quasi-Newton approaches.
- Frank-Wolfe and coordinate descent algorithms.

What we have not (or should have) covered

Import approaches and concepts

- distributed optimization.
- proximal splitting / ADMM.
- Quasi-Newton approaches.
- Frank-Wolfe and coordinate descent algorithms.

The question

Should we care that much about minimizing finite sums when all we want is minimizing an expectation?

Statistical learning basics

Statistical learning setting:

- Data $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, i.i.d. from distribution $\mathcal{D}$
- Hypothesis class (here linear) $x \mapsto \theta^\top \Phi(x)$, $\theta \in \Theta \subset \mathbb{R}^d$
- Loss function $\ell(y, \theta^\top \Phi(x))$
- **Goal**: $\min_{\theta \in \Theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(y, \theta^\top \Phi(x))]$

Statistical learning basics

Two main approaches

- **Empirical risk minimization** with batch/incremental methods

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \theta^\top \Phi(x_i)) + \Omega(\theta)$$

- Minimize **expected risk** with SGD

$$\min_{\theta \in \mathbb{R}^d} \mathbb{E}_{(x,y) \sim \mathcal{D}} [\ell(y, \theta^\top \Phi(x))]$$

- **Question**: Which is better?

Statistical learning basics

Two main approaches

- **Empirical risk minimization** with batch/incremental methods

$$\min_{\theta \in \mathbb{R}^d} \left\{ \hat{f}(\theta) := \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \theta^\top \Phi(x_i)) \right\} \text{ s.t. } \Omega(\theta) \leq D$$

- Minimize **expected risk** with SGD

$$\min_{\theta \in \mathbb{R}^d} \left\{ f(\theta) := \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(y, \theta^\top \Phi(x))] \right\}$$

- **Question**: Which is better?

Empirical Risk Minimization

$$\hat{\theta} := \arg\min_{\theta \in \Theta} \hat{f}(\theta)$$

Approximation/Estimation:

$$f(\hat{\theta}) - \min_{\theta \in \mathbb{R}^d} f(\theta) = \underbrace{f(\hat{\theta}) - \min_{\theta \in \Theta} f(\theta)}_{\text{estimation error}} + \underbrace{\min_{\theta \in \Theta} f(\theta) - \min_{\theta \in \mathbb{R}^d} f(\theta)}_{\text{approximation error}}$$

- Controlled with **regularization** (bias/variance, over/under-fitting)

Empirical Risk Minimization

$$\hat{\theta} := \arg\min_{\theta \in \Theta} \hat{f}(\theta)$$

Approximation/Estimation/Optimization:

$$f(\hat{\theta}) - \min_{\theta \in \mathbb{R}^d} f(\theta) = \underbrace{f(\hat{\theta}) - \min_{\theta \in \Theta} f(\theta)}_{\text{estimation error}} + \underbrace{\min_{\theta \in \Theta} f(\theta) - \min_{\theta \in \mathbb{R}^d} f(\theta)}_{\text{approximation error}}$$

- Controlled with **regularization** (bias/variance, over/under-fitting)
- $\hat{\theta}$ obtained *approximately* by optimization:

$$f(\tilde{\theta}) - \min_{\theta \in \mathbb{R}^d} f(\theta) = \underbrace{f(\tilde{\theta}) - f(\hat{\theta})}_{\text{optimization error}} + f(\hat{\theta}) - \min_{\theta \in \mathbb{R}^d} f(\theta)$$

- Key insight of Bottou and Bousquet (2008): no need to optimize below statistical error!

ERM: uniform convergence

- Deviations of $\hat{f}$ from f can be bounded for all $\theta \in \Theta$:

$$\mathbb{E}[\sup_{\theta \in \Theta} |\hat{f}(\theta) - f(\theta)|] \leq \frac{BD}{\sqrt{n}}$$

  - $\Theta = \{\theta : \|\theta\| \leq D\}$
  - $B = GR$ Lipschitz constant (G-Lipschitz loss, data radius R)
  - Tools from concentration of measure
- Bound estimation error ($\theta_D := \arg\min_{\theta \in \Theta} f(\theta)$):

$$\mathbb{E}[f(\hat{\theta}) - f(\theta_D)] \leq \mathbb{E}[f(\hat{\theta}) - \hat{f}(\hat{\theta}) + \underbrace{\hat{f}(\hat{\theta}) - \hat{f}(\theta_D)}_{\leq 0} + \hat{f}(\theta_D) - f(\theta_D)]$$

$$\leq 2\mathbb{E}[\sup_{\theta \in \Theta} |\hat{f}(\theta) - f(\theta)|] \leq \frac{2BD}{\sqrt{n}}$$

- Same as SGD!

ERM: fast rates

Estimation error can be smaller than $O(1/\sqrt{n})$

- $O(1/\mu n)$ for μ-strongly convex f and $\hat{f}$
  - Similar to SGD on strongly convex functions
  - Warning: large μ will increase approximation error!
- $O(1/n^\alpha)$, $\alpha \in [1/2, 1]$ by making assumptions on the data distribution $\mathcal{D}$ in classification problems:
  - Separable data $\to O(1/n)$
  - Better rate when $P(y = 1|x)$ puts little mass near 1/2

When finite sum optimization helps

- Good optimization of $\hat{f}$ helps with fast rates
- No dependence on gradient variance
- More robust to ill-conditioning, easier step-sizes
- See (Bottou and Bousquet, 2008; Babanezhad et al, 2015)
- But: SGD can do better (see work of F. Bach)

Mark the date! July 2-6th, Grenoble

Along with Naver Labs, Inria is organizing a summer school in Grenoble on artificial intelligence. Visit https://project.inria.fr/paiss/.

Among the distinguished speakers

- Lourdes Agapito (UCL)
- Kyunghyun Cho (NYU/Facebook)
- Emmanuel Dupoux (EHESS)
- Martial Hebert (CMU)
- Hugo Larochelle (Google Brain)
- Yann LeCun (Facebook/NYU)
- Jean Ponce (Inria)
- Cordelia Schmid (Inria)
- Andrew Zisserman (Oxford/Google DeepMind).
- ...

References I

A. Agarwal and L. Bottou. A lower bound for the optimization of finite sums. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *arXiv preprint arXiv:1603.05953*, 2016.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*, 2016.

Antonin Chambolle and Jérôme Darbon. On total variation minimization and surface evolution using parametric maximum flows. *International journal of computer vision*, 84(3):288, 2009.

P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *SIAM Multiscale Modeling and Simulation*, 4(4):1168–1200, 2006.

References II

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.

A. Defazio, F. Bach, and S. Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems (NIPS)*, 2014a.

A. J. Defazio, T. S. Caetano, and J. Domke. Finito: A faster, permutable incremental gradient method for big data problems. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2014b.

H. Hoefling. A path algorithm for the fused lasso signal approximator. *Journal of Computational and Graphical Statistics*, 19(4):984–1006, 2010.

R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, 12: 2297–2334, 2011.

Guanghui Lan. An optimal randomized incremental gradient method. *arXiv preprint arXiv:1507.02000*, 2015.

References III

H. Lin, J. Mairal, and Z. Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, 2015.

J. Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2): 829–855, 2015.

J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. Network flow algorithms for structured sparsity. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

Y. Nesterov. *Introductory lectures on convex optimization: a basic course*. Kluwer Academic Publishers, 2004.

Y. Nesterov. Gradient methods for minimizing composite objective function. *Mathematical Programming*, 140(1):125–161, 2013.

Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/k2). In *Doklady an SSSR*, volume 269, pages 543–547, 1983.

References IV

R. D. Nowak and M. A. T. Figueiredo. Fast wavelet-based image deconvolution using the EM algorithm. In *Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers.*, 2001.

M. Schmidt, N. Le Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *arXiv:1309.2388*, 2013.

S. Shalev-Shwartz and T. Zhang. Proximal stochastic dual coordinate ascent. *arXiv:1211.2717*, 2012.

S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, pages 1–41, 2014.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1995.

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.

L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

References V

Y. Zhang and L. Xiao. Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.