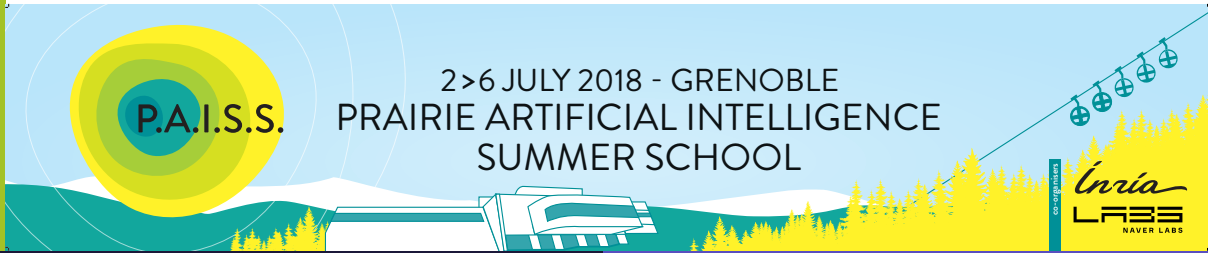
Large-Scale Optimization for Machine Learning

Julien Mairal

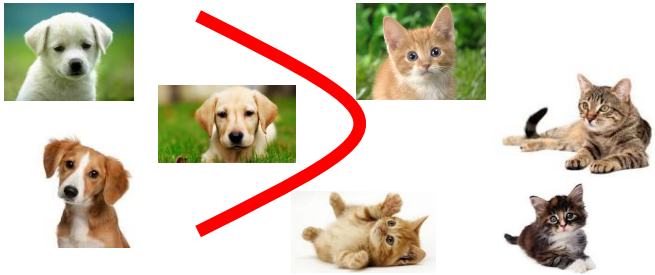
Inria Grenoble



Optimization is central to machine learning

In supervised learning, we learn a **prediction function** $h : \mathcal{X} \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\dots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

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



Optimization is central to machine learning

In supervised learning, we learn a **prediction function** $h : \mathcal{X} \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\dots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

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

The labels y_i are in

- $\{-1, +1\}$ for **binary** classification.
- $\{1, \dots, K\}$ for **multi-class** classification.
- $\mathbb{R}$ for **regression**.
- $\mathbb{R}^k$ for **multivariate regression**.
- any general set for **structured prediction**.

Optimization is central to machine learning

In supervised learning, we learn a **prediction function** $h : \mathcal{X} \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\dots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

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

Example with linear models: logistic regression, SVMs, *etc.*

- assume there exists a linear relation between y and features x in $\mathbb{R}^p$.
- $h(x) = w^\top x + b$ is parametrized by w, b in $\mathbb{R}^{p+1}$.
- L is often a **convex** loss function.
- $\Omega(h)$ is often the squared ℓ_2-norm $\|w\|^2$.

Optimization is central to machine learning

A few examples of linear models with no bias b:

Ridge regression: $$\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2}(y_i - w^\top x_i)^2 + \lambda \|w\|_2^2.$$

Linear SVM: $$\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i w^\top x_i) + \lambda \|w\|_2^2.$$

Logistic regression: $$\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y_i w^\top x_i}\right) + \lambda \|w\|_2^2.$$

Optimization is central to machine learning

In supervised learning, we learn a **prediction function** $h : \mathbb{R}^p \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\dots,n}$ with x_i in $\mathbb{R}^p$, and y_i in $\mathcal{Y}$:

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

What is specific to multilayer neural networks?

- The "neural network" space $\mathcal{H}$ is explicitly parametrized by:

$$h(x) = \sigma_k(\mathbf{A}_k \sigma_{k-1}(\mathbf{A}_{k-1} \dots \sigma_2(\mathbf{A}_2 \sigma_1(\mathbf{A}_1 x)) \dots)).$$

- Linear operations are either unconstrained or they share parameters (e.g., convolutions).
- Finding the optimal $\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_k$ yields a **non-convex** problem in **huge dimension.**

Optimization is central to machine learning

In supervised learning, we learn a **prediction function** $h : \mathbb{R}^p \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\dots,n}$ with x_i in $\mathbb{R}^p$, and y_i in $\mathcal{Y}$:

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

Even with simple linear models, it leads to challenging problems in optimization:

- **scaling** both in the problem size n and dimension p;
- being able to **exploit the problem structure** (finite sum);
- obtaining **convergence and numerical stability** guarantees;
- obtaining **statistical guarantees**.

Optimization is central to machine learning

In supervised learning, we learn a **prediction function** $h : \mathbb{R}^p \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\dots,n}$ with x_i in $\mathbb{R}^p$, and y_i in $\mathcal{Y}$:

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

It is not limited to supervised learning

$$\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(h(x_i)) + \lambda \Omega(h).$$

- L is not a classification loss any more;
- K-means, PCA, EM with mixture of Gaussian, matrix factorization, auto-encoders... can be explained with such a formulation.

Large-scale optimization for machine learning

What would be a great outline for this course

1. **Statistical learning** and empirical risk minimization.
2. General principles of **gradient-based** optimization.
   - convex optimization
   - non-convex optimization
   - non-smooth optimization
3. **Quasi-Newton** methods.
4. **Stochastic** Optimization.
5. **Distributed** Optimization.
6. ...

Large-scale optimization for machine learning

What we will do

- Introduction to statistical learning and gradient-based optimization.
- Introduction to **stochastic** optimization.
- Two advanced topics:
  - **Variance-reduced stochastic gradient descent.**
  - **Nesterov's acceleration (momentum)**.

Part I: Statistical learning and gradient-based optimization

Statistical learning

Setting

- We draw i.i.d. pairs (x_i, y_i) from some unknown distribution P.
- The objective is to minimize over all functions the **expected risk**:

$$\min_h \left\{ R(h) = \mathbb{E}_{(x,y)\sim P}[L(y, h(x))] \right\}.$$

Statistical learning

Setting

- We draw i.i.d. pairs (x_i, y_i) from some unknown distribution P.
- The objective is to minimize over all functions the **expected risk**:

$$\min_h \left\{ R(h) = \mathbb{E}_{(x,y)\sim P}[L(y, h(x))] \right\}.$$

But

Statistical learning

Setting

- We draw i.i.d. pairs (x_i, y_i) from some unknown distribution P.
- The objective is to minimize over all functions the **expected risk**:

$$\min_h \left\{ R(h) = \mathbb{E}_{(x,y)\sim P}[L(y, h(x))] \right\}.$$

But

1. we do minimize over **a class of functions** $\mathcal{H}$ only.

Statistical learning

Setting

- We draw i.i.d. pairs (x_i, y_i) from some unknown distribution P.
- The objective is to minimize over all functions the **expected risk**:

$$\min_h \left\{ R(h) = \mathbb{E}_{(x,y)\sim P}[L(y, h(x))] \right\}.$$

But

1. we do minimize over **a class of functions** $\mathcal{H}$ only.
2. datasets are often finite and we minimize instead the **empirical risk**:

$$\min_{h\in\mathcal{H}} \left\{ R_n(h) = \frac{1}{n}\sum_{i=1}^{n} [L(y_i, h(x_i))] \right\}.$$

Statistical learning

Setting

- We draw i.i.d. pairs (x_i, y_i) from some unknown distribution P.
- The objective is to minimize over all functions the **expected risk**:

$$\min_h \left\{ R(h) = \mathbb{E}_{(x,y)\sim P}[L(y, h(x))] \right\}.$$

But

1. we do minimize over **a class of functions** $\mathcal{H}$ only.
2. datasets are often finite and we minimize instead the **empirical risk**:

$$\min_{h\in\mathcal{H}} \left\{ R_n(h) = \frac{1}{n}\sum_{i=1}^{n} [L(y_i, h(x_i))] \right\}.$$

3. we minimize **approximately**.

Statistical learning

$$\hat{h}_n \in \arg\min_{h \in \mathcal{H}} R_n(h).$$

Approximation/Estimation:

$$R(\hat{h}_n) - \min_h R(h) = \underbrace{R(\hat{h}_n) - \min_{h \in \mathcal{H}} R(h)}_{\text{estimation error}} + \underbrace{\min_{h \in \mathcal{H}} R(h) - \min_h R(h)}_{\text{approximation error}}$$

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

Statistical learning

$$\hat{h}_n \in \arg\min_{h \in \mathcal{H}} R_n(h).$$

Approximation/Estimation/Optimization:

$$R(\hat{h}_n) - \min_h R(h) = \underbrace{R(\hat{h}_n) - \min_{h \in \mathcal{H}} R(h)}_{\text{estimation error}} + \underbrace{\min_{h \in \mathcal{H}} R(h) - \min_h R(h)}_{\text{approximation error}}$$

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

$$R(\tilde{h}_n) - \min_h R(h) = \underbrace{R(\tilde{h}_n) - R(\hat{h}_n)}_{\text{optimization error}} + R(\hat{h}_n) - \min_h R(h)$$

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

Statistical learning

Classical rates of estimation

- $O(D(\mathcal{H})/\sqrt{n})$ with $D(\mathcal{H})$ growing with the class of function $\mathcal{H}$.
- under specific conditions, faster rates may be achieved $O(1/n)$.

more details in `http://www.di.ens.fr/~fbach/fbach_frejus_2017.pdf`

Statistical learning

Classical rates of estimation

- $O(D(\mathcal{H})/\sqrt{n})$ with $D(\mathcal{H})$ growing with the class of function $\mathcal{H}$.
- under specific conditions, faster rates may be achieved $O(1/n)$.

more details in `http://www.di.ens.fr/~fbach/fbach_frejus_2017.pdf`

What conclusions can we draw from an optimization perspective?

- convergence rate of stochastic gradient descent (at least for convex problems) may be **asymptotically optimal**.

Statistical learning

Classical rates of estimation

- $O(D(\mathcal{H})/\sqrt{n})$ with $D(\mathcal{H})$ growing with the class of function $\mathcal{H}$.
- under specific conditions, faster rates may be achieved $O(1/n)$.

more details in `http://www.di.ens.fr/~fbach/fbach_frejus_2017.pdf`

What conclusions can we draw from an optimization perspective?

- convergence rate of stochastic gradient descent (at least for convex problems) may be **asymptotically optimal**.
- faster algorithms than SGD are not always useful, but they are if
  - they are easier to use than SGD (**no parameter tuning**).
  - if forgetting the initial condition with SGD takes time (**hard to know in advance**).

Statistical learning

Classical rates of estimation

- $O(D(\mathcal{H})/\sqrt{n})$ with $D(\mathcal{H})$ growing with the class of function $\mathcal{H}$.
- under specific conditions, faster rates may be achieved $O(1/n)$.

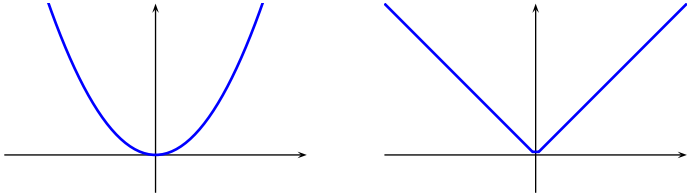
more details in `http://www.di.ens.fr/~fbach/fbach_frejus_2017.pdf`

What conclusions can we draw from an optimization perspective?

- convergence rate of stochastic gradient descent (at least for convex problems) may be **asymptotically optimal**.
- faster algorithms than SGD are not always useful, but they are if
  - they are easier to use than SGD (**no parameter tuning**).
  - if forgetting the initial condition with SGD takes time (**hard to know in advance**).
- **mathematics, engineering, and experiments are needed.**

Basics of gradient-based optimization

Smooth vs non-smooth



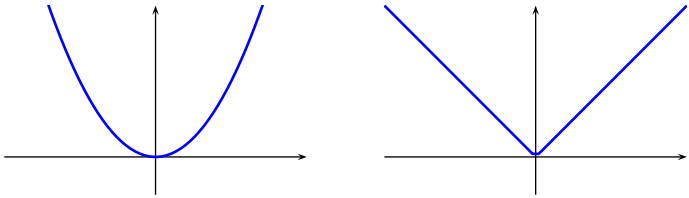
(a) smooth (b) non-smooth

An important quantity to quantify smoothness is the **Lipschitz constant** of the gradient:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|.$$

Basics of gradient-based optimization

Smooth vs non-smooth



(a) smooth (b) non-smooth

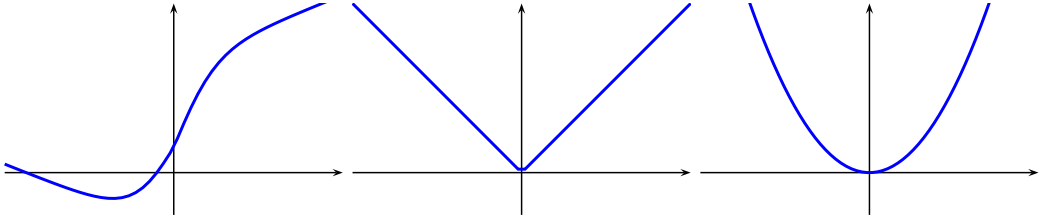
An important quantity to quantify smoothness is the **Lipschitz constant** of the gradient:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|.$$

If f is twice differentiable, L may be chosen as the **largest eigenvalue** of the Hessian $\nabla^2 f$. This is an upper-bound on the function curvature.

Basics of gradient-based optimization

Convex vs non-convex



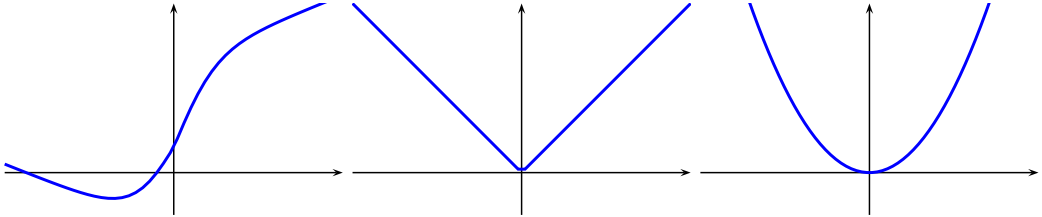
(a) non-convex (b) convex (c) strongly-convex

An important quantity to quantify convexity is the **strong-convexity** constant

$$f(x) \geq f(y) + \nabla f(y)^\top (x - y) + \frac{\mu}{2}\|x - y\|^2,$$

Basics of gradient-based optimization

Convex vs non-convex



(a) non-convex (b) convex (c) strongly-convex

An important quantity to quantify convexity is the **strong-convexity** constant

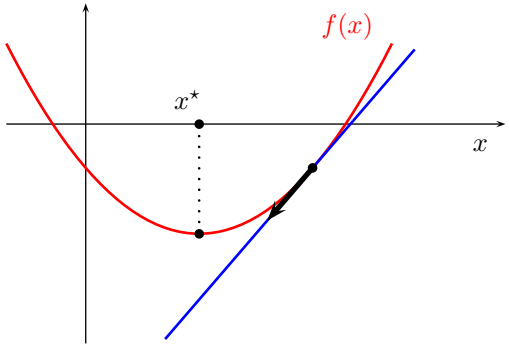
$$f(x) \geq f(y) + \nabla f(y)^\top (x - y) + \frac{\mu}{2}\|x - y\|^2,$$

If f is twice differentiable, μ may be chosen as the **smallest eigenvalue** of the Hessian $\nabla^2 f$. This is a lower-bound on the function curvature.

Basics of gradient-based optimization

Convex Functions

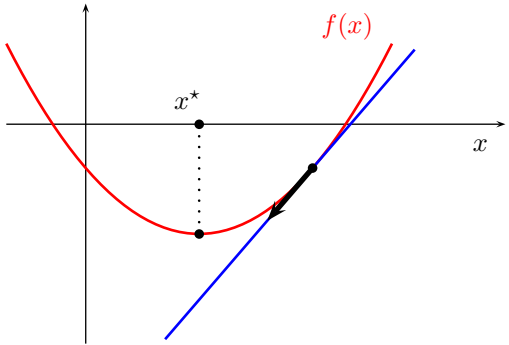
Why do we care about convexity?



Basics of gradient-based optimization

Convex Functions

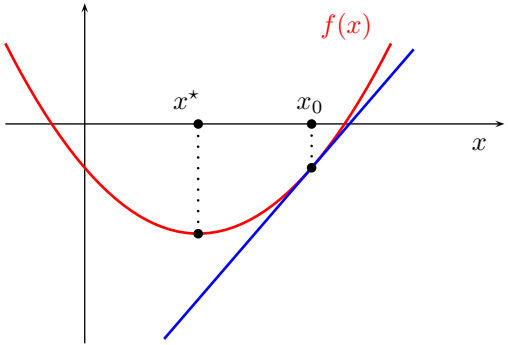
Local observations give information about the global optimum



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

Basics of gradient-based optimization

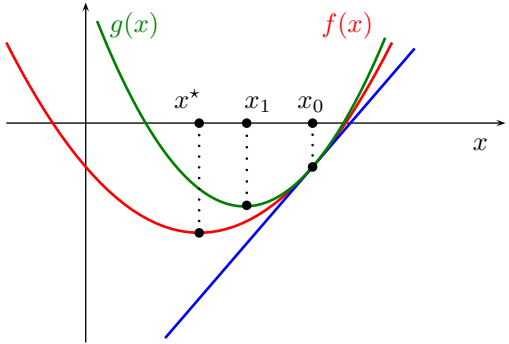
If f is convex and smooth



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

Basics of gradient-based optimization

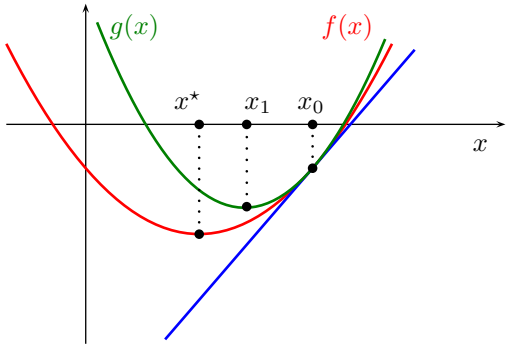
If ∇f is L-Lipschitz continuous (f does not need to be convex)



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

Basics of gradient-based optimization

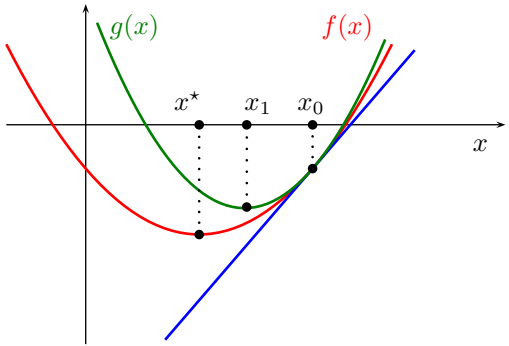
If ∇f is L-Lipschitz continuous (f does not need to be convex)



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

Basics of gradient-based optimization

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



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

Basics of gradient-based optimization

Gradient descent algorithm

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

Theorem

Consider the algorithm

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

Then,

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

Basics of gradient-based optimization

Gradient descent algorithm

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

Theorem

Consider the algorithm

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

Then,

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

How to prove this?

Read Nesterov's book! [Nesterov, 2004].

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,

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

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,

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

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,

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,

$$f(x_t) \leq g_t(x_t)$$

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,

$$f(x_t) \leq g_t(x_t) = g_t(x^\star) - \frac{L}{2}\|x^\star - x_t\|_2^2$$

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,

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

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,

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

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

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

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,

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

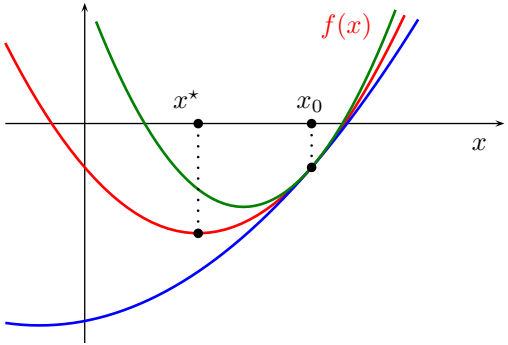
By summing from $t = 1$ to T, we have a telescopic sum

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

(green) - (red) - (blue) - telescopic sum

Basics of gradient-based optimization

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



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

Basics of gradient-based optimization

Proposition

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

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

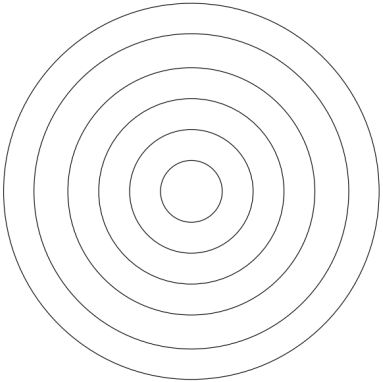
We call that a **linear** convergence rate.

Remarks

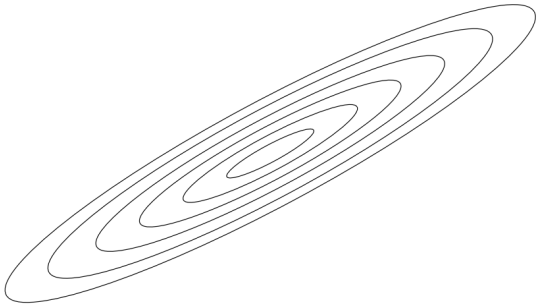
- if f is twice differentiable, L and μ represent the larget and smallest eigenvalues of the Hessian, respectively.
- L/μ is called the **condition number**.

Basics of gradient-based optimization

Picture from F. Bach



(large μ/L)



(small μ/L)

Proof

We start from a (blue) inequality from the previous proof

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

Proof

We start from a (blue) 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, blue! $f(x_t) \geq f^\star + \frac{\mu}{2}\|x_t - x^\star\|_2^2$, and thus

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

Proof

We start from a (blue) 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, blue! $f(x_t) \geq f^\star + \frac{\mu}{2}\|x_t - x^\star\|_2^2$, 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, green! $f(x_t) \leq f^\star + \nabla f(x^\star)^\top (x_t - x^\star) + \frac{L}{2}\|x_t - x^\star\|^2$ with $\nabla f(x^\star) = 0$:

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

Proof

We start from a (blue) 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, blue! $f(x_t) \geq f^\star + \frac{\mu}{2}\|x_t - x^\star\|_2^2$, 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, green! $f(x_t) \leq f^\star + \nabla f(x^\star)^\top (x_t - x^\star) + \frac{L}{2}\|x_t - x^\star\|^2$ with $\nabla f(x^\star) = 0$:

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

It is all about green and blue.

Basics of gradient-based optimization: composite problems

A **composite** optimization problem consists of minimizing the sum of a smooth and non-smooth function

$$\min_{x \in \mathbb{R}^p} \{f(x) = f_0(x) + \psi(x)\},$$

where f_0 is L-smooth and ψ is convex but not necessarily smooth.

Example

A popular choice is $\psi(x) = \|x\|_1$, which induces sparsity.

Basics of gradient-based optimization: composite problems

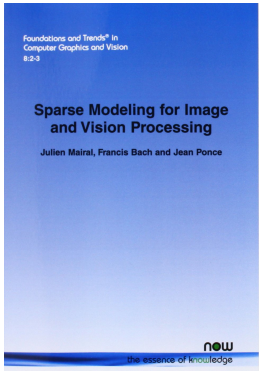
A **composite** optimization problem consists of minimizing the sum of a smooth and non-smooth function

$$\min_{x \in \mathbb{R}^p} \{f(x) = f_0(x) + \psi(x)\},$$

where f_0 is L-smooth and ψ is convex but not necessarily smooth.

Example

A popular choice is $\psi(x) = \|x\|_1$, which induces sparsity.



J. Mairal, F. Bach and J. Ponce. *Sparse Modeling for Image and Vision Processing*. Foundations and Trends in Computer Graphics and Vision. 2014.

Basics of gradient-based optimization: composite problems

Remark: with stepsize $1/L$, gradient descent may be interpreted as iteratively minimizing a tight upper-bound:

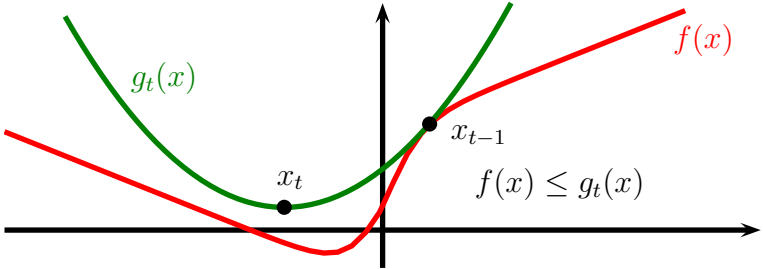
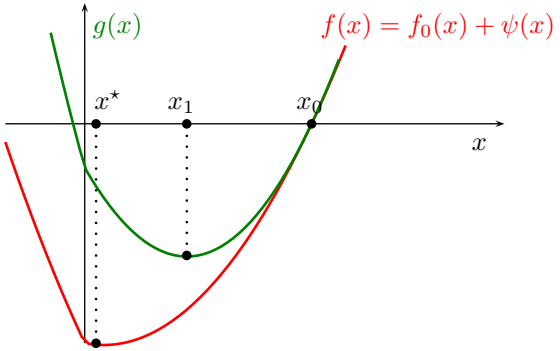


Figure: At each step, we update $x_t \in \arg\min_{x \in \mathbb{R}^p} g_t(x)$

Basics of gradient-based optimization: composite problems

An important inequality for composite functions

If ∇f_0 is L-Lipschitz continuous



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

Basics of gradient-based optimization: composite problems

Gradient descent for minimizing f consists of

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

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.

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

Basics of gradient-based optimization: composite problems

Remarks

- also known as **forward-backward** algorithm;
- same convergence rates as GD - same proofs;
- there exists **line search schemes** to automatically tune L;
- proximal operator can be computed for many interesting functions.

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]...

Accelerated gradient descent methods

Nesterov introduced in the 80's an acceleration scheme for the gradient descent algorithm.

Generalization to the composite setting: FISTA

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

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

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

- $f(x_t) - f^\star = O(1/t^2)$ 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

The complexity to guarantee $f(x_t) - f^\star \leq \varepsilon$, is given below

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

Remarks

- the rate of FISTA is optimal for a "first-order local black box" [Nesterov, 2004].
- for non-convex problems, acceleration often works in practice, but is poorly understood from a theoretical perspective (local convexity? convexity along trajectories? saddle-point escape?).

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 facts

- 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

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

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. $f(x_t) \leq \varphi_t^\star \stackrel{\Delta}{=} \min_{x \in \mathbb{R}^p} \varphi_t(x)$.

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, $\lambda_t = \prod_{t=1}^t (1-\alpha_t)$.

Part II: Stochastic optimization and variance reduction

Stochastic optimization

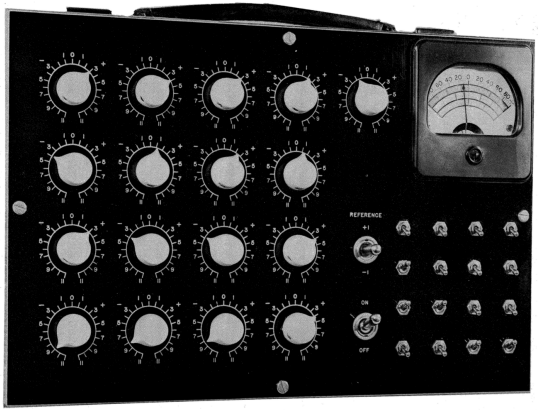


Figure: Adaline, [Widrow and Hoff, 1960]: A physical device that performs **least square regression using stochastic gradient descent**.

Problems considered in this part

Minimization of (large) finite sums

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

Minimization of expectations with infinite data

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

The finite-sum problem corresponds to the empirical risk minimization problem, whereas the second one corresponds to the **expected cost**.

The stochastic 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 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).
- Due to Robbins and Monro [1951].

[Nemirovski, Juditsky, Lan, and Shapiro, 2009, Moulines and Bach, 2011]...

The stochastic gradient descent algorithm

What theory tells us

- first use a **constant step-size**: the objective function value decreases quickly (as full GD) until it oscillates.
- then, use a **decreasing step size** and start **averaging**.

The stochastic gradient descent algorithm

What theory tells us

- first use a **constant step-size**: the objective function value decreases quickly (as full GD) until it oscillates.
- then, use a **decreasing step size** and start **averaging**.

What practice "seems" to tell us

- for deep networks, reducing twice the learning rate by 10 every x epochs seems ok.
- use a mini batch (cheap parallelization), but not too large?
- use Nesterov/Heavy-ball's extrapolation?
- use an adaptive learning rate strategy? (see next slide)
- averaging? or not?
- solutions tend to have small norm: implicit regularization?

The stochastic gradient descent algorithm

What theory tells us

- first use a **constant step-size**: the objective function value decreases quickly (as full GD) until it oscillates.
- then, use a **decreasing step size** and start **averaging**.

What practice "seems" to tell us

- for deep networks, reducing twice the learning rate by 10 every x epochs seems ok.
- use a mini batch (cheap parallelization), but not too large?
- use Nesterov/Heavy-ball's extrapolation?
- use an adaptive learning rate strategy? (see next slide)
- averaging? or not?
- solutions tend to have small norm: implicit regularization?

Practice changes every year. Beware of big inductive claims.

The stochastic gradient descent algorithm

Inspired by Jamie Soel's presentation at NIPS'2018

- SGD:

$$x_t = x_{t-1} - \eta_t \nabla f_t(x_{t-1}).$$

The stochastic gradient descent algorithm

Inspired by Jamie Soel's presentation at NIPS'2018

- SGD:

$$x_t = x_{t-1} - \eta_t \nabla f_t(x_{t-1}).$$

- Heavy-Ball momentum:

$$x_t = x_{t-1} - \eta_t \nabla f_t(x_{t-1}) + \beta_t (x_{t-1} - x_{t-2}).$$

The stochastic gradient descent algorithm

Inspired by Jamie Soel's presentation at NIPS'2018

- SGD:

$$x_t = x_{t-1} - \eta_t \nabla f_t(x_{t-1}).$$

- Heavy-Ball momentum:

$$x_t = x_{t-1} - \eta_t \nabla f_t(x_{t-1}) + \beta_t(x_{t-1} - x_{t-2}).$$

- Nesterov's extrapolation:

$$x_t = x_{t-1} - \eta_t \nabla f_t(x_{t-1} + \beta_t(x_{t-1} - x_{t-2})) + \beta_t(x_{t-1} - x_{t-2}).$$

The stochastic gradient descent algorithm

Inspired by Jamie Soel's presentation at NIPS'2018

- SGD:

$$x_t = x_{t-1} - \eta_t \nabla f_t(x_{t-1}).$$

- Heavy-Ball momentum:

$$x_t = x_{t-1} - \eta_t \nabla f_t(x_{t-1}) + \beta_t(x_{t-1} - x_{t-2}).$$

- Nesterov's extrapolation:

$$x_t = x_{t-1} - \eta_t \nabla f_t(x_{t-1} + \beta_t(x_{t-1} - x_{t-2})) + \beta_t(x_{t-1} - x_{t-2}).$$

- AdaGrad [Duchi et al., 2011]

$$x_t = x_{t-1} - \eta_t H_t^{-1} \nabla f_t(x_{t-1}).$$

The stochastic gradient descent algorithm

Inspired by Jamie Soel's presentation at NIPS'2018

- SGD:

$$x_t = x_{t-1} - \eta_t \nabla f_t(x_{t-1}).$$

- Heavy-Ball momentum:

$$x_t = x_{t-1} - \eta_t \nabla f_t(x_{t-1}) + \beta_t(x_{t-1} - x_{t-2}).$$

- Nesterov's extrapolation:

$$x_t = x_{t-1} - \eta_t \nabla f_t(x_{t-1} + \beta_t(x_{t-1} - x_{t-2})) + \beta_t(x_{t-1} - x_{t-2}).$$

- AdaGrad [Duchi et al., 2011]

$$x_t = x_{t-1} - \eta_t H_t^{-1} \nabla f_t(x_{t-1}).$$

- Adam [Kingma and Ba, 2014]:

$$x_t = x_{t-1} - \eta_t H_t^{-1} \nabla f_t(x_{t-1}) + \beta_t H_t^{-1} H_{t-1}(x_{t-1} - x_{t-2}).$$

Back to finite sums

Consider now the case of interest:

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

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 $\mathbb{E}[f(x_k) - f^\star] \leq \varepsilon$ is

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

Incremental gradient descent methods

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

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

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.
- SVRG is better than FISTA if $n \geq \sqrt{L/\mu}$.

Incremental gradient descent methods

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

inspired from F. Bach's slides.

Variance reduction

Consider two random variables X, Y and define

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

Then,

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

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

Incremental gradient descent methods

inspired from F. Bach's slides.

Variance reduction

Consider two random variables X, Y and define

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

Then,

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

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 **larger 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

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

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

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

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	$O\left(\max\left(n, \frac{L}{\mu}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$
Accelerated versions	$\tilde{O}\left(\max\left(n, \sqrt{n\frac{L}{\mu}}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$

- Acceleration for specific algorithms [Shalev-Shwartz and Zhang, 2014, Lan, 2015, Allen-Zhu, 2016].
- Generic acceleration: Catalyst [Lin, Mairal, and Harchaoui, 2015].
- see [Agarwal and Bottou, 2015] for discussions about optimality.
- SVRG is better than FISTA if $n \geq \sqrt{L/\mu}$.
- AccSVRG is better than SVRG if $n \leq L/\mu$.

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	$\tilde{O}\left(\max\left(n, \sqrt{n\frac{\bar{L}}{\mu}}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$

- if n is huge (one-pass learning): use SGD!

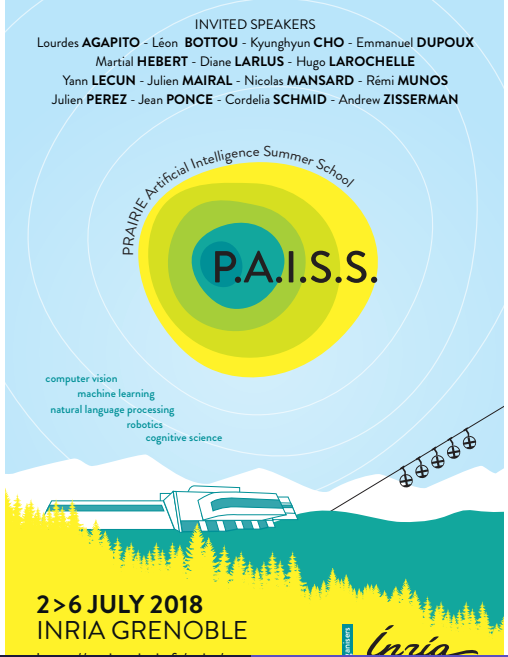
Questions about incremental methods

Do they work in practice?

- for convex objectives
  - on **training** error: huge improvements over well-tuned SGD.
  - on **test** error: less clear (not worse than SGD).
  - **much easier** to use than SGD.
- for non-convex objectives: nothing clear yet.

When is acceleration useful?

- when the problem is badly conditioned (L/μ large).
- when the amount of data is large, but not too large (such that one-pass un-regularized SGD does not work).


INVITED SPEAKERS
Lourdes AGAPITO - Léon BOTTOU - Kyunghyun CHO - Emmanuel DUPOUX
Martial HEBERT - Diane LARLUS - Hugo LAROCHELLE
Yann LECUN - Julien MAIRAL - Nicolas MANSARD - Rémi MUNOS
Julien PEREZ - Jean PONCE - Cordelia SCHMID - Andrew ZISSERMAN
PRAIRIE Artificial Intelligence Summer School
P.A.I.S.S.
computer vision
machine learning
natural language processing
robotics
cognitive science
2>6 JULY 2018
INRIA GRENOBLE

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.

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

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.

References II

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.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

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.

Eric Moulines and Francis R Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pages 451–459, 2011.

References III

Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4): 1574–1609, 2009.

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.

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.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

References IV

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.

B. Widrow and M. E. Hoff. Adaptive switching circuits. In *IRE WESCON convention record*, volume 4, pages 96–104. New York, 1960.

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.

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.