

Optimization for Machine Learning

CS-439

Lecture 10: Accelerated Gradient Descent, Gradient-free, and Applications

Martin Jaggi

EPFL – github.com/epfml/OptML_course

May 17, 2024

Chapter X.1

Accelerated Gradient Descent

Smooth convex functions: less than $\mathcal{O}(1/\varepsilon)$ steps?

Fixing L and $R = \|\mathbf{x}_0 - \mathbf{x}^*\|$, the error of gradient descent after T steps is $\mathcal{O}(1/T)$.

Lee and Wright [LW19]:

- ▶ A better upper bound of $o(1/T)$ holds.
- ▶ A lower bound of $\Omega(1/T^{1+\delta})$ also holds, for any fixed $\delta > 0$.

So, gradient descent is **slightly** faster on smooth functions than what we proved, but not significantly.

First-order methods: less than $\mathcal{O}(1/\varepsilon)$ steps?

Maybe gradient descent is not the best possible algorithm?

After all, it is just **some** algorithm that uses gradient information.

First-order method:

- ▶ An algorithm that gains access to f only via an oracle that is able to return values of f and ∇f at arbitrary points.
- ▶ Gradient descent is a specific first-order method.

What is the **best** first-order method for smooth convex functions, the one with the smallest upper bound on the number of oracle calls in the worst case?

Nemirovski and Yudin 1979 [NY83]: **every** first-order method needs in the worst case $\Omega(1/\sqrt{\varepsilon})$ steps (gradient evaluations) in order to achieve an additive error of ε on smooth functions.

There is a gap between $\mathcal{O}(1/\varepsilon)$ (gradient descent) and the lower bound!

Acceleration for smooth convex functions: $\mathcal{O}(1/\sqrt{\varepsilon})$ steps

Nesterov 1983 [Nes83, Nes18]: There is a first-order method that needs only $\mathcal{O}(1/\sqrt{\varepsilon})$ steps on smooth convex functions, and by the lower bound of Nemirovski and Yudin, this is a best possible algorithm!

The algorithm is known as (Nesterov's) accelerated gradient descent.

A number of (similar) optimal algorithms with other proofs of the $\mathcal{O}(1/\sqrt{\varepsilon})$ upper bound are known, but there is no well-established “simplest proof”.

Here: a recent proof based on [potential functions](#) [BG17]. Proof is simple but not very instructive (it works, but it's not clear why).

Nesterov's accelerated gradient descent

Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be convex, differentiable, and smooth with parameter L . Choose $\mathbf{z}_0 = \mathbf{y}_0 = \mathbf{x}_0$ arbitrary. For $t \geq 0$, set

$$\begin{aligned}\mathbf{y}_{t+1} &:= \mathbf{x}_t - \frac{1}{L} \nabla f(\mathbf{x}_t) \\ \mathbf{z}_{t+1} &:= \mathbf{z}_t - \frac{t+1}{2L} \nabla f(\mathbf{x}_t) \\ \mathbf{x}_{t+1} &:= \frac{t+1}{t+3} \mathbf{y}_{t+1} + \frac{2}{t+3} \mathbf{z}_{t+1}.\end{aligned}$$

- ▶ Perform a “smooth step” from \mathbf{x}_t to \mathbf{y}_{t+1} .
- ▶ Perform a more aggressive step from \mathbf{z}_t to \mathbf{z}_{t+1} .
- ▶ Next iterate \mathbf{x}_{t+1} is a weighted average of \mathbf{y}_{t+1} and \mathbf{z}_{t+1} , where we compensate for the more aggressive step by giving \mathbf{z}_{t+1} a relatively low weight.

Why should this work??

Nesterov's accelerated gradient descent: Error bound

Theorem

Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be convex and differentiable with a global minimum \mathbf{x}^* ; furthermore, suppose that f is smooth with parameter L . Accelerated gradient descent yields

$$f(\mathbf{y}_T) - f(\mathbf{x}^*) \leq \frac{2L \|\mathbf{z}_0 - \mathbf{x}^*\|^2}{T(T+1)}, \quad T > 0.$$

To reach error at most ε , accelerated gradient descent therefore only needs $\mathcal{O}(1/\sqrt{\varepsilon})$ steps instead of $\mathcal{O}(1/\varepsilon)$.

Recall the bound for gradient descent:

$$f(\mathbf{x}_T) - f(\mathbf{x}^*) \leq \frac{L}{2T} \|\mathbf{x}_0 - \mathbf{x}^*\|^2, \quad T > 0.$$

Nesterov's accelerated gradient descent: The potential function

Idea: assign a **potential** $\Phi(t)$ to each time t and show that $\Phi(t+1) \leq \Phi(t)$.

Out of the blue: let's define the potential as

$$\Phi(t) := t(t+1) (f(\mathbf{y}_t) - f(\mathbf{x}^*)) + 2L \|\mathbf{z}_t - \mathbf{x}^*\|^2.$$

If we can show that the potential always decreases, we get

$$\underbrace{T(T+1) (f(\mathbf{y}_T) - f(\mathbf{x}^*)) + 2L \|\mathbf{z}_T - \mathbf{x}^*\|^2}_{\Phi(T)} \leq \underbrace{2L \|\mathbf{z}_0 - \mathbf{x}^*\|^2}_{\Phi(0)}.$$

Rewriting this, we get the claimed error bound.

(optional material) Potential function decrease: Three Ingredients

Sufficient decrease for the smooth step from \mathbf{x}_t to \mathbf{y}_{t+1} :

$$f(\mathbf{y}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{2L} \|\nabla f(\mathbf{x}_t)\|^2; \quad (1)$$

Vanilla analysis for the more aggressive step from \mathbf{z}_t to \mathbf{z}_{t+1} : ($\gamma = \frac{t+1}{2L}$, $\mathbf{g}_t = \nabla f(\mathbf{x}_t)$):

$$\mathbf{g}_t^\top (\mathbf{z}_t - \mathbf{x}^*) = \frac{t+1}{4L} \|\mathbf{g}_t\|^2 + \frac{L}{t+1} (\|\mathbf{z}_t - \mathbf{x}^*\|^2 - \|\mathbf{z}_{t+1} - \mathbf{x}^*\|^2); \quad (2)$$

Convexity (graph of f is above the tangent hyperplane at \mathbf{x}_t):

$$f(\mathbf{x}_t) - f(\mathbf{w}) \leq \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{w}), \quad \mathbf{w} \in \mathbb{R}^d. \quad (3)$$

(optional material) Potential function decrease: Proof

By definition of potential,

$$\begin{aligned}\Phi(t+1) &= t(t+1)(f(\mathbf{y}_{t+1}) - f(\mathbf{x}^*)) + 2(t+1)(f(\mathbf{y}_{t+1}) - f(\mathbf{x}^*)) + 2L\|\mathbf{z}_{t+1} - \mathbf{x}^*\|^2, \\ \Phi(t) &= t(t+1)(f(\mathbf{y}_t) - f(\mathbf{x}^*)) + 2L\|\mathbf{z}_t - \mathbf{x}^*\|^2.\end{aligned}$$

Now, prove that $\Delta := (\Phi(t+1) - \Phi(t))/(t+1) \leq 0$:

$$\begin{aligned}\Delta &= t(f(\mathbf{y}_{t+1}) - f(\mathbf{y}_t)) + 2(f(\mathbf{y}_{t+1}) - f(\mathbf{x}^*)) + \frac{2L}{t+1} \left(\|\mathbf{z}_{t+1} - \mathbf{x}^*\|^2 - \|\mathbf{z}_t - \mathbf{x}^*\|^2 \right) \\ &\stackrel{(2)}{=} t(f(\mathbf{y}_{t+1}) - f(\mathbf{y}_t)) + 2(f(\mathbf{y}_{t+1}) - f(\mathbf{x}^*)) + \frac{t+1}{2L} \|\mathbf{g}_t\|^2 - 2\mathbf{g}_t^\top(\mathbf{z}_t - \mathbf{x}^*) \\ &\stackrel{(1)}{\leq} t(f(\mathbf{x}_t) - f(\mathbf{y}_t)) + 2(f(\mathbf{x}_t) - f(\mathbf{x}^*)) - \frac{1}{2L} \|\mathbf{g}_t\|^2 - 2\mathbf{g}_t^\top(\mathbf{z}_t - \mathbf{x}^*) \\ &\leq t(f(\mathbf{x}_t) - f(\mathbf{y}_t)) + 2(f(\mathbf{x}_t) - f(\mathbf{x}^*)) - 2\mathbf{g}_t^\top(\mathbf{z}_t - \mathbf{x}^*) \\ &\stackrel{(3)}{\leq} t\mathbf{g}_t^\top(\mathbf{x}_t - \mathbf{y}_t) + 2\mathbf{g}_t^\top(\mathbf{x}_t - \mathbf{x}^*) - 2\mathbf{g}_t^\top(\mathbf{z}_t - \mathbf{x}^*) \\ &= \mathbf{g}_t^\top((t+2)\mathbf{x}_t - t\mathbf{y}_t - 2\mathbf{z}_t) \stackrel{(\text{algo})}{=} \mathbf{g}_t^\top \mathbf{0} = 0. \quad \square\end{aligned}$$

Chapter X.2

Zero-Order Optimization

Look mom no gradients!

Can we optimize $\min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x})$ if without access to gradients?

meet the newest fanciest optimization algorithm,...

Random search

pick a random direction $\mathbf{d}_t \in \mathbb{R}^d$

$\gamma := \operatorname{argmin}_{\gamma \in \mathbb{R}} f(\mathbf{x}_t + \gamma \mathbf{d}_t)$ (line-search)

$\mathbf{x}_{t+1} := \mathbf{x}_t + \gamma \mathbf{d}_t$

Convergence rate for derivative-free random search

Theorem: Converges same as gradient descent - up to a slow-down factor d .

Proof. Assume that f is a L -smooth convex, differentiable function. For any γ , by smoothness, we have:

$$f(\mathbf{x}_t + \gamma \mathbf{d}_t) \leq f(\mathbf{x}_t) + \gamma \mathbf{d}_t^\top \nabla f(\mathbf{x}_t) + \frac{\gamma^2 L}{2} \|\mathbf{d}_t\|^2$$

Minimizing the upper bound, there is a step size $\bar{\gamma}$ for which

$$f(\mathbf{x}_t + \bar{\gamma} \mathbf{d}_t) \leq f(\mathbf{x}_t) - \frac{1}{L} \left(\frac{\mathbf{d}_t^\top}{\|\mathbf{d}_t\|} \nabla f(\mathbf{x}_t) \right)^2$$

The step size γ we actually took (based on f directly) can only be better:

$$f(\mathbf{x}_t + \gamma \mathbf{d}_t) \leq f(\mathbf{x}_t + \bar{\gamma} \mathbf{d}_t) .$$

Taking expectations, and using the Lemma $\mathbb{E}_{\mathbf{r}}(\mathbf{r}^\top \mathbf{g})^2 = \frac{1}{d} \|\mathbf{g}\|^2$ for $\mathbf{r} \sim \text{sphere} \subseteq \mathbb{R}^d$:

$$\mathbb{E}[f(\mathbf{x}_t + \gamma \mathbf{d}_t)] \leq \mathbb{E}[f(\mathbf{x}_t)] - \frac{1}{Ld} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|^2] .$$

Convergence rate for derivative-free random search

Same as what we obtained for **gradient descent**,
now with an **extra factor of d** . d can be huge!!!

Can do the same for different function classes, as before

- ▶ For convex functions, we get a rate of $\mathcal{O}(dL/\varepsilon)$.
- ▶ For strongly convex, we get $\mathcal{O}(dL/\mu \log(1/\varepsilon))$.

Always d times the complexity of gradient descent on the function class.

credits to Moritz Hardt

Applications for derivative-free random search

Applications

- ▶ can be used for **Reinforcement learning**
- ▶ memory and communication advantages: never need to store a gradient
- ▶ hyperparameter optimization, and other difficult e.g. discrete optimization problems

Reinforcement learning

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t, \mathbf{e}_t).$$

where \mathbf{s}_t is the **state** of the system, \mathbf{a}_t is the control **action**, and \mathbf{e}_t is some random **noise**. We assume that f is fixed, but unknown.

We search for a control 'policy'

$$\mathbf{a}_t := \pi(\mathbf{a}_1, \dots, \mathbf{a}_{t-1}, \mathbf{s}_0, \dots, \mathbf{s}_t).$$

which takes a trajectory of the dynamical system and outputs a new control action.
Want to maximize overall **reward**

$$\begin{aligned} \max_{\mathbf{a}_t} \mathbb{E}_{\mathbf{e}_t} \left[\sum_{t=0}^N R_t(\mathbf{s}_t, \mathbf{a}_t) \right] \\ \text{s.t. } \mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t, \mathbf{e}_t) \\ (\mathbf{s}_0 \text{ given}) \end{aligned}$$

Examples: Simulations, Games (e.g. Atari), Alpha Go

Chapter X.3

Adaptive & other SGD Methods

Momentum SGD

Momentum variant of SGD (Polyak, 1964)

pick a stochastic gradient \mathbf{g}_t

$$\mathbf{m}_t := \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t \quad (\text{momentum term})$$

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma \mathbf{m}_t$$

(standard choice of $\mathbf{g}_t := \nabla f_j(\mathbf{x}_t)$ for sum-structured objective functions $f = \sum_j f_j$)

- ▶ momentum from previous gradients
- ▶ is a variant of the Nesterov acceleration seen before
- ▶ key element of deep learning optimizers, necessary for top accuracy

Adagrad

Adagrad is an adaptive variant of SGD

pick a stochastic gradient \mathbf{g}_t

$$\text{update } [G_t]_i := \sum_{s=0}^t ([\mathbf{g}_s]_i)^2 \quad \forall i$$

$$[\mathbf{x}_{t+1}]_i := [\mathbf{x}_t]_i - \frac{\gamma}{\sqrt{[G_t]_i}} [\mathbf{g}_t]_i \quad \forall i$$

- ▶ chooses an **adaptive, coordinate-wise** learning rate
- ▶ strong performance in practice
- ▶ Variants: Adadelta, Adam, RMSprop

Adam

Adam is a momentum variant of Adagrad

pick a stochastic gradient \mathbf{g}_t

$$\mathbf{m}_t := \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t \quad (\text{momentum term})$$

$$[\mathbf{v}_t]_i := \beta_2 [\mathbf{v}_{t-1}]_i + (1 - \beta_2) ([\mathbf{g}_t]_i)^2 \quad \forall i \quad (\text{2nd-order statistics})$$

$$[\mathbf{x}_{t+1}]_i := [\mathbf{x}_t]_i - \frac{\gamma}{\sqrt{[\mathbf{v}_t]_i}} [\mathbf{m}_t]_i \quad \forall i$$

- ▶ faster forgetting of older weights
- ▶ momentum from previous gradients (see acceleration)
- ▶ (simplified version, without correction for initialization of $\mathbf{m}_0, \mathbf{v}_0$)
- ▶ strong performance in practice, e.g. for self-attention networks

SignSGD

Only use the sign (one bit) of each gradient entry:

SignSGD is a communication efficient variant of SGD.

pick a stochastic gradient \mathbf{g}_t

$$[\mathbf{x}_{t+1}]_i := [\mathbf{x}_t]_i - \gamma_t \text{sign}([\mathbf{g}_t]_i) \quad \forall i$$

(with possible rescaling of γ_t with $\|\mathbf{g}_t\|_1$)

- ▶ communication efficient for distributed training
- ▶ convergence issues

ClippedSGD






Clip the gradients to a predefined maximum length $c > 0$.

$$\begin{aligned} &\text{pick a stochastic gradient } \mathbf{g}_t \\ \mathbf{x}_{t+1} &:= \mathbf{x}_t - \gamma_t \text{clip}_c(\mathbf{g}_t) \quad \forall i \end{aligned}$$

with $\text{clip}_c(\mathbf{g}) := \max\left(1, \frac{c}{\|\mathbf{g}\|}\right) \cdot \mathbf{g}$

- ▶ used to avoid instabilities in deep learning training (such as LLMs)
- ▶ used with differential privacy (adding noise of comparable magnitude)
- ▶ convergence non-trivial

Bibliography

-  Nikhil Bansal and Anupam Gupta.
Potential-function proofs for first-order methods.
CoRR, abs/1712.04581, 2017.
-  Ching-Pei Lee and Stephen Wright.
First-order algorithms converge faster than $o(1/k)$ on convex problems.
In *ICML - International Conference on Machine Learning*, 2019.
-  Yurii Nesterov.
A method of solving a convex programming problem with convergence rate $o(1/k^2)$.
Soviet Math. Dokl., 27(2), 1983.
-  Yurii Nesterov.
Lectures on Convex Optimization, volume 137.
Springer, 2018.
-  Arkady. S. Nemirovsky and D. B. Yudin.
Problem complexity and method efficiency in optimization.