10-425/625: Introduction to Convex Optimization **(Fall 2023)**

Lecture 20: Momentum & Nesterov Acceleration

Instructor:[1] *Matt Gormley* *November 15, 2023*

20.1 Momentum and Acceleration for SGD

20.1.1 SGD and Mini-batch SGD

Assume now that we are trying to minimize a function f with the structure:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x).$$

Stochastic gradient descent repeatedly updates:

$$\begin{aligned} g^t &= \nabla f_{i^t}(x^t) \\ x^{t+1} &= x^t - \eta_t g^t \end{aligned}$$

where at each iteration we choose an index i^t uniformly at random from $\{1, \dots, n\}$.

Mini-batch SGD instead chooses subsets $I_t \subset \{1, \dots, n\}$ of size m (say), and computes:

$$\begin{aligned} g^t &= \frac{1}{m}\sum_{i \in I_t} \nabla f_i(x^t) \\ x^{t+1} &= x^t - \eta_t g^t \end{aligned}$$

If the subsets are chosen uniformly at random from $\{1, \dots, n\}$ then this is a valid stochastic gradient. It has a variance which is a factor of m smaller (but can be m times more expensive to compute).

[1]These notes were originally written by Ryan Tibshirani for 10-725 Fall 2019 (original version: here) and were edited and adapted for 10-425/625.

20.1.2 SGD with Classical Momentum

Here we consider SGD with classical momentum. The idea is that we include an additional weight $\beta \in [0,1]$ that trades off between how much to step in the direction of the current gradient (small β) and how much to continue moving the direction that we have been moving (large β). In short, $v^{(t)}$ keeps an exponential moving average of the gradient vectors.

$$
\begin{aligned}
g^{(t)} &= \nabla f_{i^t}(x^t) \qquad\qquad \text{or } g^{(t)} = \frac{1}{m}\sum_{i \in I_t} \nabla f_i(x^t) \\
v^{(t)} &= \beta v^{(t-1)} + (1-\beta) g^{(t)} \\
x^{(t)} &= x^{(t-1)} - v^{(t)}
\end{aligned}
$$

We implement this within the IGM with Random Permutations setup, shuffling the order in which we visit each f_i at the start of each epoch—or using mini-Batch SGD.

In practice, it's common to use two hyperparameters: β affects the terminal velocity and η is a learning rate.

$$
\begin{aligned}
v^{(t)} &= \beta v^{(t-1)} + \eta g^{(t)} \\
x^{(t)} &= x^{(t-1)} - v^{(t)}
\end{aligned}
$$

Example 20.1 (Linear Objective)**.** Suppose we had a linear objective such that the gradient was constant at every timestep: $g^{(t)} = g$. In this case, we will build momentum in the direction of $-g$ until we reach a maximum velocity with a step size of $\frac{\eta||g||_2}{1-\beta}$ where the quantity $\frac{1}{1-\beta}$ bounds how much momentum we can build up.

Physics Motivation We can think of the momentum term $v^{(t)}$ as representing the velocity of a particle where the force applied to that particle is in the direction of the negative gradient $-g^{(t)}$. The other force is viscous drag in the direction $-v^{(t)}$. Note that this differs from turbulent drag which would have force proportional to the square of the velocity or from dry friction which has constant magnitude. Writing out the partial differential equations for this system and applying Euler's method to solve it gives rise to the momentum algorithm.

So the analogy we could consider is a force $-g^{(t)}$ pulling our particle along according to some function f through water where it experiences viscous drag.

20.1.3 SGD with Nesterov Momentum

We can take Nesterov's accellerated gradient method and apply it to SGD as well. The result looks quite similar to classical momentum with an important difference: we evaluate the gradient at the the point that we would step to if we continued in the current momentum direction. The algorithm then repeats the following for $t = 1, 2, 3...$:

$$\begin{aligned} \tilde{x}^t &= x^{(t-1)} - \eta v^{(t-1)} \\ g^{(t)} &= \nabla f_{i^t}(\tilde{x}^t) \qquad\qquad \text{or } g^{(t)} = \frac{1}{m}\sum_{i \in I_t} \nabla f_i(\tilde{x}^t) \\ v^{(t)} &= \beta v^{(t-1)} + \eta g^{(t)} \\ x^{(t)} &= x^{(t-1)} - \eta v^{(t)} \end{aligned}$$

Sutskever et al. (2013) explain the difference geometrically as shown below, where $\beta = \mu$ is the velocity parameter and $\eta = 1$.

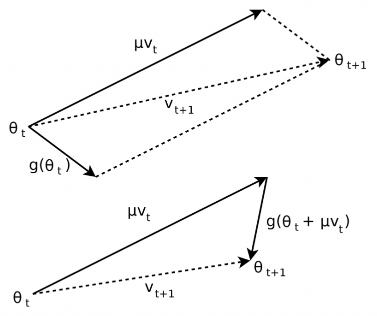


Figure 1. **(Top)** Classical Momentum **(Bottom)** Nesterov Accelerated Gradient

20.1.4 Analysis

Unfortunately, neither SGD with momentum, nor SGD with Nesterov momentum enjoys the $O(1/k^2)$ convergence rate that we saw for Nesterov accelleration with the proximal gradient method. We are instead left with

$O(1/k)$ convergence rate because the stochastic gradients here nullify the advantage.

20.2 Proximal Gradient Descent + Nesterov Acceleration

Recall from a previous lecture

20.2.1 The Algorithm

The particular problem class we'll focus on now will be: we want to minimize an unconstrained function f which can be written as the sum of a "nice", convex, function g and a potentially non-smooth convex function h, i.e.

$$\min_{x \in \mathbb{R}^d} g(x) + h(x).$$

The prox. GD algorithm alternates the following steps:

1. We compute $y^{t+1} = x^t - \eta_t \nabla g(x^t)$.
2. We then compute our next iterate by solving:

$$x^{t+1} = \arg\min_{x \in \mathbb{R}^d} \left[h(x) + \frac{1}{2\eta_t} \|x - y^{t+1}\|_2^2 \right].$$

We can equivalently write the update by defining the proximal operator,

$$\text{prox}_f(v) = \arg\min_x \left(f(x) + \frac{1}{2} \|x - v\|_2^2 \right).$$

and then for each time step we update:

$$x^{t+1} = \text{prox}_{\eta_t h}(x^t - \eta_t \nabla g(x^t))$$

20.2.2 Nesterov acceleration for proximal gradient descent

Acceleration Turns out we can accelerate proximal gradient descent in order to achieve the optimal $O(1/\sqrt{\epsilon})$ (i.e. $O(1/k^2)$) convergence rate. Four ideas (three acceleration methods) by Nesterov:

- 1983: original acceleration idea for smooth functions
- 1988: another acceleration idea for smooth functions
- 2005: smoothing techniques for nonsmooth functions, coupled with original acceleration idea
- 2007: acceleration idea for composite functions[2]

We will follow Beck and Teboulle (2008), an extension of Nesterov (1983) to composite functions[3]

Accelerated proximal gradient method As before, consider:

$$\min_x \; g(x) + h(x)$$

where g convex, differentiable, and h convex. Accelerated proximal gradient method: choose initial point $x^{(0)} = x^{(-1)} \in \mathbb{R}^n$, repeat:

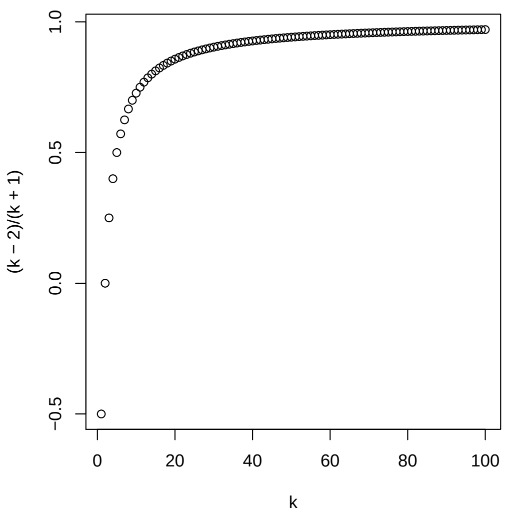
$$\begin{aligned} v &= x^{(t-1)} + \frac{t-2}{t+1}(x^{(t-1)} - x^{(t-2)}) \\ x^{(t)} &= \operatorname{prox}_{\eta_t h}\big(v - \eta_t \nabla g(v)\big) \end{aligned}$$

for $t = 1, 2, 3, \ldots$

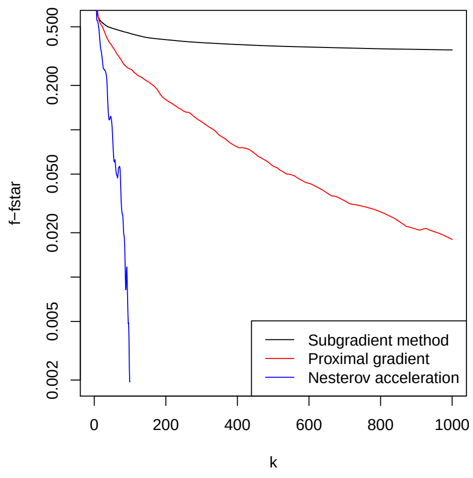
- First step $t = 1$ is just usual proximal gradient update
- After that, $v = x^{(t-1)} + \frac{t-2}{t+1}(x^{(t-1)} - x^{(t-2)})$ carries some "momentum" from previous iterations
- When $h = 0$ we get accelerated gradient method

[2]Each step uses entire history of previous steps and makes two prox calls
[3]Each step uses information from two last steps and makes one prox call

Momentum weights:



Back to lasso example: acceleration can really help!



Note: accelerated proximal gradient is not a descent method

Backtracking line search Backtracking under with acceleration in different ways. Simple approach: fix $\beta < 1$, $\eta_0 = 1$. At iteration k, start with $\eta = \eta_{k-1}$, and while

$$g(x^+) > g(v) + \nabla g(v)^T (x^+ - v) + \frac{1}{2\eta} \|x^+ - v\|_2^2$$

shrink $\eta = \beta\eta$, and let $x^+ = \text{prox}_{\eta h}(v - \eta\nabla g(v))$. Else keep x^+

Note that this strategy forces us to take decreasing step sizes ... (more complicated strategies exist which avoid this)

Convergence analysis For criterion $f(x) = g(x) + h(x)$, we assume as before:

- g is convex, differentiable, $\text{dom}(g) = \mathbb{R}^n$, and ∇g is Lipschitz continuous with constant $L > 0$
- h is convex, $\text{prox}_{\eta h}(x) = \text{argmin}_z\{\|x - z\|_2^2/(2\eta) + h(z)\}$ can be evaluated

Theorem: Accelerated proximal gradient method with fixed step size $\eta \le 1/L$ satisfies

$$f(x^{(k)}) - f^\star \le \frac{2\|x^{(0)} - x^\star\|_2^2}{\eta(k+1)^2}$$

and same result holds for backtracking, with η replaced by β/L

Achieves optimal rate $O(1/k^2)$ or $O(1/\sqrt{\epsilon})$ for first-order methods

FISTA Back to lasso problem:

$$\min_\beta \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1$$

Recall ISTA (Iterative Soft-thresholding Algorithm):

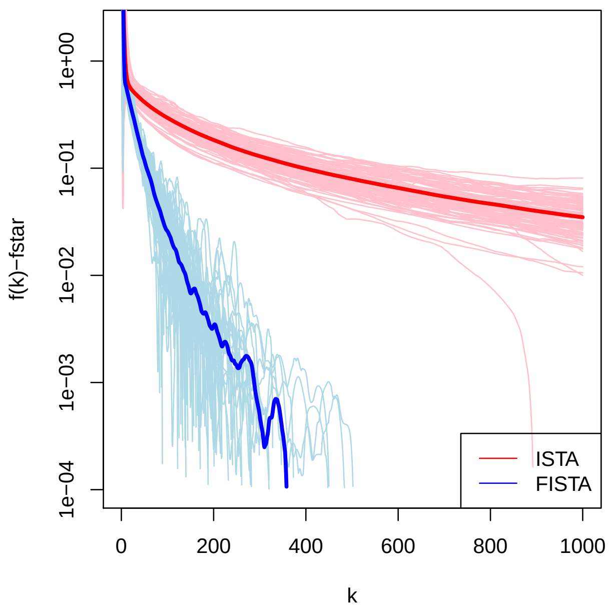
$$\beta^{(t)} = S_{\lambda\eta_t}(\beta^{(t-1)} + \eta_t X^T(y - X\beta^{(t-1)})), \quad t = 1, 2, 3, \dots$$

$S_\lambda(\cdot)$ being vector soft-thresholding. Applying acceleration gives us FISTA (F is for Fast):[4] for $t = 1, 2, 3, \dots,$

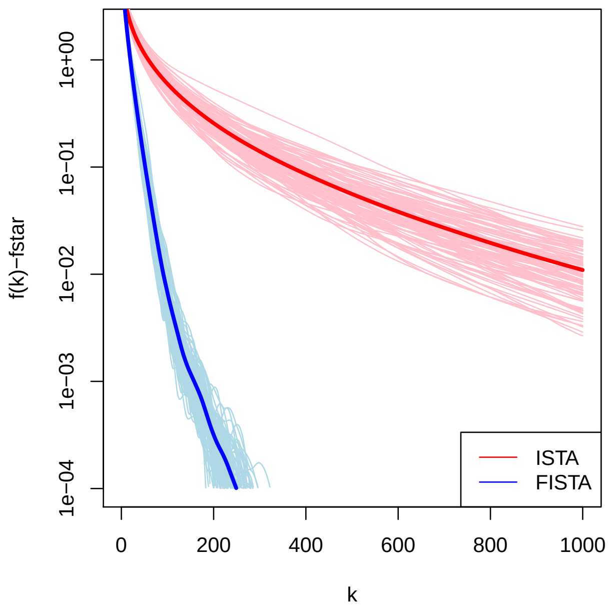
$$v = \beta^{(t-1)} + \frac{t-2}{t+1}(\beta^{(t-1)} - \beta^{(t-2)})$$
$$\beta^{(t)} = S_{\lambda\eta_t}\big(v + \eta_t X^T(y - Xv)\big),$$

[4]Beck and Teboulle (2008) actually call their general acceleration technique (for general g, h) FISTA, which may be somewhat confusing

Lasso regression: 100 instances (with $n = 100$, $p = 500$):



Lasso logistic regression: 100 instances ($n = 100$, $p = 500$):



Is acceleration always useful?

Acceleration can be a very effective speedup tool ... but should it always be used?

In practice the speedup of using acceleration is diminished in the presence of warm starts. For example, suppose want to solve lasso problem for tuning parameters values

$$\lambda_1 > \lambda_2 > \cdots > \lambda_r$$

- When solving for λ_1, initialize $x^{(0)} = 0$, record solution $\hat{x}(\lambda_1)$
- When solving for λ_j, initialize $x^{(0)} = \hat{x}(\lambda_{j-1})$, the recorded solution for λ_{j-1}

Over a fine enough grid of λ values, proximal gradient descent can often perform just as well without acceleration

Sometimes backtracking and acceleration can be disadvantageous! For the matrix completion problem: the proximal gradient update is

$$B^+ = S_\lambda\Big(B + t\big(P_\Omega(Y) - P^\perp(B)\big)\Big)$$

where S_λ is the matrix soft-thresholding operator ... requires SVD

- One backtracking loop evaluates prox, across various values of t. For matrix completion, this means multiple SVDs ...
- Acceleration changes argument we pass to prox: $v - t\nabla g(v)$ instead of $x - t\nabla g(x)$. For matrix completion (and $t = 1$),

$$B - \nabla g(B) = \underbrace{P_\Omega(Y)}_{\text{sparse}} + \underbrace{P_\Omega^\perp(B)}_{\text{low rank}} \quad \Rightarrow \quad \text{fast SVD}$$

$$V - \nabla g(V) = \underbrace{P_\Omega(Y)}_{\text{sparse}} + \underbrace{P_\Omega^\perp(V)}_{\substack{\text{not necessarily} \\ \text{low rank}}} \quad \Rightarrow \quad \text{slow SVD}$$