# Acceleration

## SIE 449/549: Optimization for Machine Learning

Afrooz Jalilzadeh

Department of Systems and Industrial Engineering
University of Arizona

# First-order Method

- Consider the following optimization problem

$$\min_x \ f(x)$$

  where $f : \mathbb{R}^n \to \mathbb{R}$ is convex and **differentiable**

- Gradient descent: choose initial $x_0 \in \mathbb{R}^n$, repeat:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

- If $\nabla f(x)$ is Lipschitz with constant $L$ and $\alpha = 1/L$, then:

$$f(x_k) - f(x^*) \leq \frac{L}{2k} \|x_0 - x^*\|^2 = \mathcal{O}(1/k)$$

**Question:** Is $\mathcal{O}(1/k)$ the optimal rate, among methods that only use gradient information, like gradient descent?

# Gradient Method

- Consider a smooth (differentiable), convex function $f$, with $dom(f) = \mathbb{R}^n$, such that $\nabla f$ Lipschitz continuous with constant $L > 0$
- Given an initial point $x_0 \in \mathbb{R}^n$, we are allowed to use any algorithm that produces iterates satisfying:

$$x_{k+1} \in x_0 + span\{\nabla f(x_0), \ldots, \nabla f(x_k)\}$$

- We call such an algorithm a *(smooth) first-order* method
- We can lower bound the convergence rate of first-order methods by $1/k^2$:

$$\frac{3L\|x_0 - x^*\|^2}{32(k+1)^2} \leq f(x_k) - f(x^*)$$

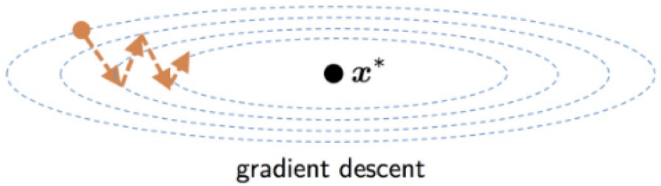
**Question:** Can we achieve the lower bound with a practical first-order method?

# Polyak's momentum

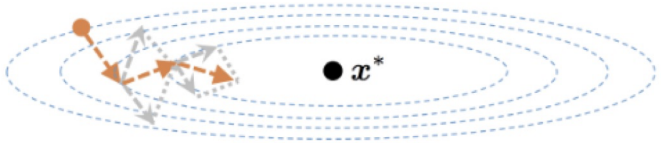
**Momentum gradient descent**, or the **heavy-ball algorithm**

- Idea: Combines the current gradient with a history of the previous step to accelerate the convergence of the algorithm
- Introduces a *momentum* term $\beta(x_k - x_{k-1})$, where $\beta$ is a hyperparameter

$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1})$$




gradient descent




heavy-ball method

# Nesterov's Accelerated Gradient Method

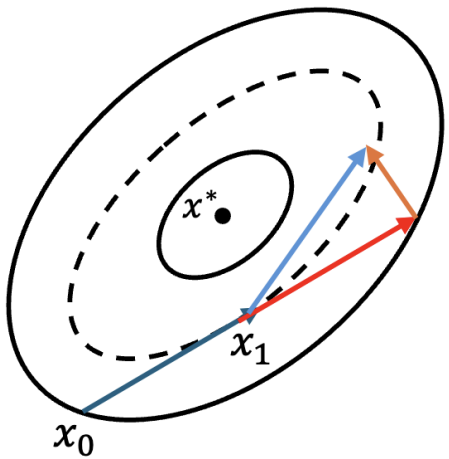
- Gap: Heavy-ball method was shown to achieve the lower bounds for some class of problems. Can we change the method to achieve rate of $\mathcal{O}(1/k^2)$ for first-order methods under convexity and $L-$smoothness assumption?
- The Heavy-ball method:

$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1})$$

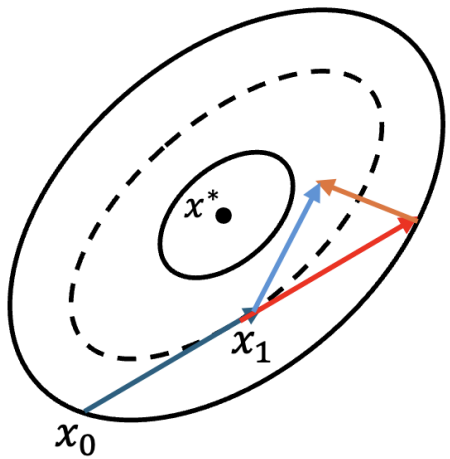
- Can be equivalently written as a two-step procedure:

- In Heavy-ball, we compute the gradient of $f$ at $x_k$
- What if we compute the gradient at a point that looks *more similar* to the motions we perform, even after the gradient calculation in heavy-ball?
- Nesterov's suggestion:

# Heavy ball vs Nesterov's Acceleration




$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1})$$




$$x_{k+1} = x_k - \alpha \nabla f(x_k + \beta(x_k - x_{k-1})) + \beta(x_k - x_{k-1})$$

# Nesterov's Accelerated Gradient Method

**Algorithm 1** Nesterov's Accelerated Gradient Method

**Initialization**: pick $x_0 \in \mathbb{R}^n$ arbitrarily, let $x_{-1} = x_0$
**for** $k = 0, 1, 2, \ldots$ **do**

$$y = x_k + \frac{k-2}{k+1}(x_k - x_{k-1})$$
$$x_{k+1} = y - \alpha \nabla f(y)$$

**end for**

- In the definition of $y$, the second term $\frac{k-2}{k+1}(x_k - x_{k-1})$ is like a momentum term which continues to push us in the direction pointing from $x_{k-1}$ to $x_k$
- The weight $\frac{k-2}{k+1}$ gets closer and closer to 1 as k gets larger, this term helps accelerate the convergence of the algorithm when it is close to the optimum
- If $f$ is smooth and convex with $L$-Lipschitz continuous gradient, then the accelerated gradient method with a fixed step size $\alpha = 1/L$ satisfies

$$f(x_k) - f(x^*) \leq \frac{2L\|x_0 - x^*\|^2}{(k+1)^2}$$

# Accelerated Proximal Gradient Method, aka FISTA

$$\min_x f(x) \triangleq g(x) + h(x)$$

- $g$ is convex, differentiable, $dom(g) = \mathbb{R}^n$, and $\nabla g$ is Lipschitz continuous with L
- $h$ is convex and its proximal map can be easily computed

Proximal Gradient Method: $x_{k+1} = prox_{h,\alpha}\left(x_k - \alpha \nabla g(x_k)\right)$

**Algorithm 2** FISTA

**Initialization**: pick $x_0 \in \mathbb{R}^n$ arbitrarily, $t_0 = 1$, $y_0 = x_0$
**for** $k = 0, 1, 2, \ldots$ **do**

$$x_{k+1} = prox_{h,\alpha}\left(y_k - \alpha \nabla g(y_k)\right)$$

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$$

$$y_{k+1} = x_{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right)(x_{k+1} - x_k)$$

**end for**

- $\alpha = 1/L$ then convergence rate $f(x_k) - f(x^*) \leq \frac{2L\|x_0 - x^*\|^2}{(k+1)^2}$

# Application: Image deblurring

- Consider the following linear inverse problem:

$$Ax = b + w$$

  where $A \in \mathbb{R}^{m \times m}$, $b \in \mathbb{R}^m$ are known, $w$ is an unknown noise (or perturbation) vector, and $x$ is the “true" and unknown signal/image to be estimated

- In image blurring problems, for example, $b \in \mathbb{R}^m$ represents the blurred image, and $x \in \mathbb{R}^m$ is the unknown true image

- Both $b$ and $x$ are formed by stacking the columns of their corresponding two-dimensional image

- The matrix $A$ describes the blur operator

- The problem of estimating $x$ from the observed blurred and noisy image $b$ is called an image deblurring problem

# Application: Image deblurring

- Minimize the data error:
$$\min_x \|Ax - b\|_2$$
- In many applications, such as image deblurring, $A$ is ill-conditioned $\left(\frac{\lambda_{max}}{\lambda_{min}} >> 1\right)$, hence the solution usually has a huge norm and is thus meaningless

$\Longrightarrow$ regularization methods are required to stabilize the solution

- Most images have a sparse representation $\Longrightarrow$ $\ell_1$ regularization

- Image deblurring problem can be solved by Proximal Gradient and also FISTA*

*Beck, Amir, and Marc Teboulle. “A fast iterative shrinkage-thresholding algorithm for linear inverse problems." SIAM journal on imaging sciences 2.1 (2009): 183-202.

# Application: Image deblurring

Figure 1: Proximal gradient for 100 and 200 iterations

Figure 2: FISTA for 100 and 200 iterations