Mathematics of Data: From Theory to Computation

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 3: Optimality of Convergence rates. Accelerated/Stochastic Gradient Descent

Laboratory for Information and Inference Systems (LIONS)
École Polytechnique Fédérale de Lausanne (EPFL)

EE-556 (Fall 2021)

License Information for Mathematics of Data Slides

Recall: Gradient descent

Problem (Unconstrained convex problem)

Consider the following convex minimization problem:

$$f^\star = \min_{\mathbf{x} \in \mathbb{R}^p} f(\mathbf{x})$$

- f *is a convex function that is*
  - *proper* : $\forall \mathbf{x} \in \mathbb{R}^p$, $-\infty < f(\mathbf{x})$ *and there exists* $\mathbf{x} \in \mathbb{R}^p$ *such that* $f(x) < +\infty$.
  - *closed* : *The epigraph* epi$f$ $= \{(\mathbf{x}, t) \in \mathbb{R}^{p+1}, f(\mathbf{x}) \leq t\}$ *is closed.*
  - *smooth* : f *is differentiable and its gradient* ∇f *is* L*-Lipschitz.*
- *The solution set* $\mathcal{S}^\star := \{\mathbf{x}^\star \in \operatorname{dom}(f) : f(\mathbf{x}^\star) = f^\star\}$ *is nonempty.*

Gradient descent (GD)

Choose a starting point $\mathbf{x}^0$ and iterate

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k)$$

where α_k is a step-size to be chosen so that $\mathbf{x}^k$ converges to $\mathbf{x}^\star$.

Convergence rate of gradient descent

Theorem

Let f be a twice-differentiable convex function, if

$$
\begin{array}{lllll}
f \text{ is } L\text{-smooth}, & \alpha = \frac{1}{L}: & f(\mathbf{x}^k) - f(\mathbf{x}^\star) & \leq \dfrac{2L}{k+4} & \|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2 \\
f \text{ is } L\text{-smooth and } \mu\text{-strongly convex}, & \alpha = \frac{2}{L+\mu}: & \|\mathbf{x}^k - \mathbf{x}^\star\|_2 & \leq \left(\dfrac{L-\mu}{L+\mu}\right)^{k} & \|\mathbf{x}^0 - \mathbf{x}^\star\|_2 \\
f \text{ is } L\text{-smooth and } \mu\text{-strongly convex}, & \alpha = \frac{1}{L}: & \|\mathbf{x}^k - \mathbf{x}^\star\|_2 & \leq \left(\dfrac{L-\mu}{L+\mu}\right)^{\frac{k}{2}} & \|\mathbf{x}^0 - \mathbf{x}^\star\|_2
\end{array}
$$

Note that $\frac{L-\mu}{L+\mu} = \frac{\kappa-1}{\kappa+1}$, where $\kappa := \frac{L}{\mu}$ is the condition number of $\nabla^2 f$.

Information theoretic lower bounds [20]

What is the **best** achievable rate for a **first-order** method?

$f \in \mathcal{F}_L^\infty$: ∞-differentiable and L-smooth

It is possible to construct a function in $\mathcal{F}_L^\infty$, for which **any** first order method must satisfy

$$f(\mathbf{x}^k) - f(\mathbf{x}^\star) \geq \frac{3L}{32(k+1)^2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2 \quad \text{for all } k \leq (p-1)/2$$

$f \in \mathcal{F}_{L,\mu}^\infty$: ∞-differentiable, L-smooth and μ-strongly convex

It is possible to construct a function in $\mathcal{F}_{L,\mu}^\infty$, for which **any** first order method must satisfy

$$\|\mathbf{x}^k - \mathbf{x}^\star\|_2 \geq \left(\frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_2$$

Gradient descent is $O(1/k)$ for $\mathcal{F}_L^\infty$ and it is slower for $\mathcal{F}_{L,\mu}^\infty$, hence it does not achieve the lower bounds!

Accelerated gradient descent algorithm

Problem

Is it possible to design first-order methods with convergence rates matching the theoretical lower bounds?

Accelerated gradient descent algorithm

Problem

Is it possible to design first-order methods with convergence rates matching the theoretical lower bounds?

Solution [Nesterov's accelerated scheme]

Accelerated Gradient Descent (AGD) methods achieve optimal convergence rates.

Accelerated gradient descent algorithm

Problem

Is it possible to design first-order methods with convergence rates matching the theoretical lower bounds?

Solution [Nesterov's accelerated scheme]

Accelerated Gradient Descent (AGD) methods achieve optimal convergence rates.

Accelerated Gradient algorithm for L-smooth (AGD-L)

1. Set $\mathbf{x}^0 = \mathbf{y}^0 \in \mathrm{dom}\,(f)$ and $t_0 := 1$.
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} \mathbf{x}^{k+1} & = \mathbf{y}^k - \frac{1}{L}\nabla f(\mathbf{y}^k) \\ t_{k+1} & = (1 + \sqrt{4t_k^2 + 1})/2 \\ \mathbf{y}^{k+1} & = \mathbf{x}^{k+1} + \frac{(t_k - 1)}{t_{k+1}}(\mathbf{x}^{k+1} - \mathbf{x}^k) \end{cases}$$

Accelerated gradient descent algorithm

Problem

Is it possible to design first-order methods with convergence rates matching the theoretical lower bounds?

Solution [Nesterov's accelerated scheme]

Accelerated Gradient Descent (AGD) methods achieve optimal convergence rates.

Accelerated Gradient algorithm for L-smooth (AGD-L)

1. Set $\mathbf{x}^0 = \mathbf{y}^0 \in \mathrm{dom}\,(f)$ and $t_0 := 1$.
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} \mathbf{x}^{k+1} &= \mathbf{y}^k - \frac{1}{L}\nabla f(\mathbf{y}^k) \\ t_{k+1} &= (1 + \sqrt{4t_k^2 + 1})/2 \\ \mathbf{y}^{k+1} &= \mathbf{x}^{k+1} + \frac{(t_k - 1)}{t_{k+1}}(\mathbf{x}^{k+1} - \mathbf{x}^k) \end{cases}$$

Accelerated Gradient algorithm for L-smooth and μ-strongly convex (AGD-μL)

1. Choose $\mathbf{x}^0 = \mathbf{y}^0 \in \mathrm{dom}\,(f)$
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} \mathbf{x}^{k+1} &= \mathbf{y}^k - \frac{1}{L}\nabla f(\mathbf{y}^k) \\ \mathbf{y}^{k+1} &= \mathbf{x}^{k+1} + \alpha(\mathbf{x}^{k+1} - \mathbf{x}^k) \end{cases}$$

where $\alpha = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$.

Accelerated gradient descent algorithm

Problem

Is it possible to design first-order methods with convergence rates matching the theoretical lower bounds?

Solution [Nesterov's accelerated scheme]

Accelerated Gradient Descent (AGD) methods achieve optimal convergence rates.

Accelerated Gradient algorithm for L-smooth (AGD-L)

1. Set $\mathbf{x}^0 = \mathbf{y}^0 \in \mathrm{dom}\,(f)$ and $t_0 := 1$.
2. For $k = 0, 1, \ldots$, iterate

$$\begin{cases} \mathbf{x}^{k+1} &= \mathbf{y}^k - \frac{1}{L}\nabla f(\mathbf{y}^k) \\ t_{k+1} &= (1 + \sqrt{4t_k^2 + 1})/2 \\ \mathbf{y}^{k+1} &= \mathbf{x}^{k+1} + \frac{(t_k - 1)}{t_{k+1}}(\mathbf{x}^{k+1} - \mathbf{x}^k) \end{cases}$$

Accelerated Gradient algorithm for L-smooth and μ-strongly convex (AGD-μL)

1. Choose $\mathbf{x}^0 = \mathbf{y}^0 \in \mathrm{dom}\,(f)$
2. For $k = 0, 1, \ldots$, iterate

$$\begin{cases} \mathbf{x}^{k+1} &= \mathbf{y}^k - \frac{1}{L}\nabla f(\mathbf{y}^k) \\ \mathbf{y}^{k+1} &= \mathbf{x}^{k+1} + \alpha(\mathbf{x}^{k+1} - \mathbf{x}^k) \end{cases}$$

where $\alpha = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$.

Remark: ◦ AGD is not monotone, but the cost-per-iteration is essentially the same as GD.

Global convergence of AGD [20]

Theorem (f is convex with Lipschitz gradient)

If f is L-smooth or L-smooth and μ-strongly convex, the sequence $\{\mathbf{x}^k\}_{k\geq 0}$ generated by **AGD-L** *satisfies*

$$f(\mathbf{x}^k) - f^\star \leq \frac{4L}{(k+2)^2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2, \ \forall k \geq 0. \tag{1}$$

Global convergence of AGD [20]

Theorem (f is convex with Lipschitz gradient)

If f is L-smooth or L-smooth and μ-strongly convex, the sequence $\{\mathbf{x}^k\}_{k\geq 0}$ generated by **AGD-L** *satisfies*

$$f(\mathbf{x}^k) - f^\star \leq \frac{4L}{(k+2)^2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2, \ \forall k \geq 0. \tag{1}$$

AGD-L is **optimal** *for L-smooth but NOT for L-smooth and μ-strongly convex!*

Theorem (f is strongly convex with Lipschitz gradient)

If f is L-smooth and μ-strongly convex, the sequence $\{\mathbf{x}^k\}_{k\geq 0}$ generated by **AGD-μL** *satisfies*

$$f(\mathbf{x}^k) - f^\star \leq L\left(1 - \sqrt{\frac{\mu}{L}}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2, \ \forall k \geq 0 \tag{2}$$

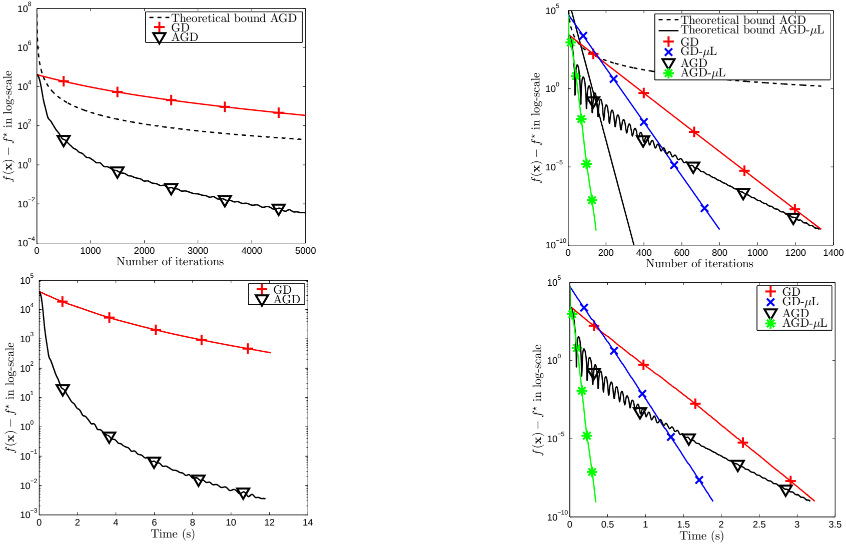
$$\|\mathbf{x}^k - \mathbf{x}^\star\|_2 \leq \sqrt{\frac{2L}{\mu}}\left(1 - \sqrt{\frac{\mu}{L}}\right)^{\frac{k}{2}} \|\mathbf{x}^0 - \mathbf{x}^\star\|_2, \ \forall k \geq 0. \tag{3}$$

Observations:
- AGD-L's iterates are not guaranteed to converge.
- AGD-L does not have a **linear** convergence rate for L-smooth and μ-strongly convex.
- AGD-μL does, but needs to know μ.
- AGD achieves the iteration lowerbound within a constant!

Example: Ridge regression

Case 1: $n = 500, p = 2000, \rho = 0$

Case 2: $n = 500, p = 2000, \rho = 0.01\lambda_p(\mathbf{A}^T\mathbf{A})$



Gradient descent vs. Accelerated gradient descent

Assumptions, step sizes and convergence rates

Gradient descent:

$$f \text{ is } L\text{-smooth}, \quad \alpha = \frac{1}{L}: \qquad f(\mathbf{x}^k) - f(\mathbf{x}^\star) \leq \frac{2L}{k+4}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2.$$

Accelerated Gradient Descent:

$$f \text{ is } L\text{-smooth}, \quad \alpha = \frac{1}{L}: \qquad f(\mathbf{x}^k) - f(x^\star) \leq \frac{4L}{(k+2)^2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2, \ \forall k \geq 0.$$

Gradient descent vs. Accelerated gradient descent

Assumptions, step sizes and convergence rates

Gradient descent:

$$f \text{ is } L\text{-smooth}, \quad \alpha = \frac{1}{L}: \qquad f(\mathbf{x}^k) - f(\mathbf{x}^\star) \leq \frac{2L}{k+4}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2.$$

Accelerated Gradient Descent:

$$f \text{ is } L\text{-smooth}, \quad \alpha = \frac{1}{L}: \qquad f(\mathbf{x}^k) - f(x^\star) \leq \frac{4L}{(k+2)^2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2, \ \forall k \geq 0.$$

Observations:

- We require α_t to be a function of L.
- It may not be possible to know exactly the Lipschitz constant.
- Adaptation to local geometry $\rightarrow$ may lead to larger steps.

Adaptive first-order methods and ⋆Newton method

Adaptive methods

Adaptive methods converge with fast rates without knowing the smoothness constant.

They do so by making use of the information from gradients and their norms.

Adaptive first-order methods and *Newton method

Adaptive methods

Adaptive methods converge with fast rates without knowing the smoothness constant.

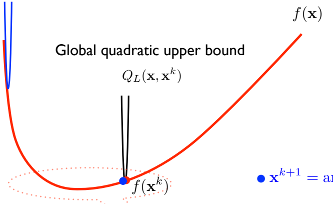
They do so by making use of the information from gradients and their norms.

*Newton method

Higher-order information, e.g., Hessian, gives a finer characterization of local behavior.

Newton method achieves asymptotically better local rates, but for additional cost.

How can we better adapt to the local geometry?

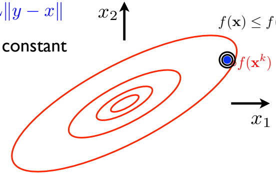


- $\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$

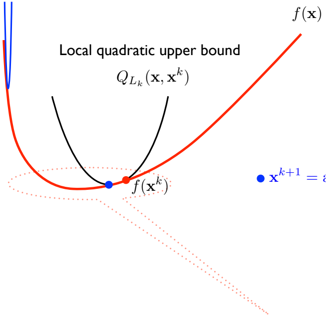
$$\|\nabla f(x) - \nabla f(y)\| \leq L\|y - x\|$$

L is a global worst-case constant

$$f(\mathbf{x}) \leq f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) + \frac{L}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2$$



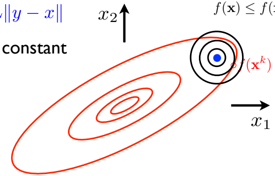
How can we better adapt to the local geometry?



- $\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L_k}{2} \|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$

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

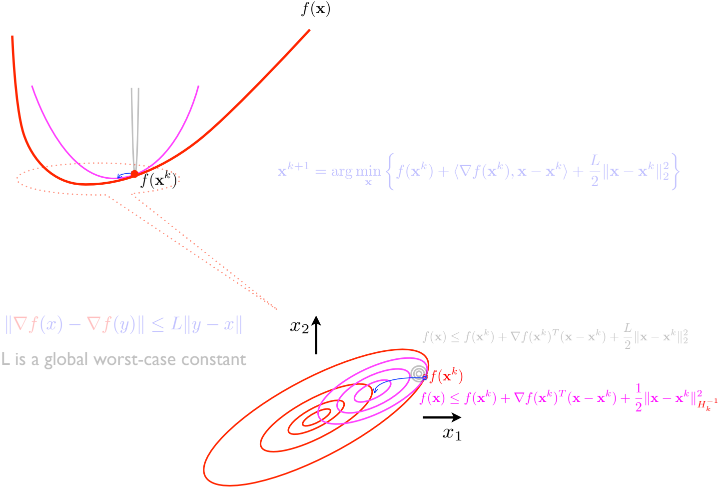
L is a global worst-case constant



$$f(\mathbf{x}) \leq f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) + \frac{L_k}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2$$

applies only locally

How can we better adapt to the local geometry?



Variable metric gradient descent algorithm

Variable metric gradient descent algorithm

1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ as a starting point and $\mathbf{H}_0 \succ 0$.
2. For $k = 0, 1, \cdots$, perform:

$$\begin{cases} \mathbf{d}^k & := -\mathbf{H}_k^{-1}\nabla f(\mathbf{x}^k), \\ \mathbf{x}^{k+1} & := \mathbf{x}^k + \alpha_k \mathbf{d}^k, \end{cases}$$

where $\alpha_k \in (0, 1]$ is a given step size.

3. Update $\mathbf{H}_{k+1} \succ 0$ if necessary.

Variable metric gradient descent algorithm

Variable metric gradient descent algorithm

1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ as a starting point and $\mathbf{H}_0 \succ 0$.
2. For $k = 0, 1, \cdots,$ perform:

$$\begin{cases} \mathbf{d}^k & := -\mathbf{H}_k^{-1} \nabla f(\mathbf{x}^k), \\ \mathbf{x}^{k+1} & := \mathbf{x}^k + \alpha_k \mathbf{d}^k, \end{cases}$$

where $\alpha_k \in (0, 1]$ is a given step size.

3. Update $\mathbf{H}_{k+1} \succ 0$ if necessary.

Common choices of the variable metric $\mathbf{H}_k$

- $\mathbf{H}_k := \lambda_k \mathbf{I}$ $\Longrightarrow$ gradient descent method.
- $\mathbf{H}_k := \mathbf{D}_k$ (a positive diagonal matrix) $\Longrightarrow$ adaptive gradient methods.
- $\mathbf{H}_k := \nabla^2 f(\mathbf{x}^k)$ $\Longrightarrow$ Newton method.
- $\mathbf{H}_k \approx \nabla^2 f(\mathbf{x}^k)$ $\Longrightarrow$ quasi-Newton method.

Adaptive gradient methods

Intuition

Adaptive gradient methods adapt locally by setting $\mathbf{H}_k$ as a function of past gradient information.

Adaptive gradient methods

Intuition

Adaptive gradient methods adapt locally by setting $\mathbf{H}_k$ as a function of past gradient information.

- Roughly speaking, $\mathbf{H}_k = \text{function}(\nabla f(\mathbf{x}^1), \nabla f(\mathbf{x}^2), \cdots, \nabla f(\mathbf{x}^k))$
- Some well-known examples:

AdaGrad [9]

$$\mathbf{H}_k = \sqrt{\sum_{t=1}^{k} (\nabla f(\mathbf{x}^t)^\top \nabla f(\mathbf{x}^t))}$$

***RmsProp [27]**

$$\mathbf{H}_k = \sqrt{\beta \mathbf{H}_{k-1} + (1-\beta)\mathrm{diag}(\nabla f(\mathbf{x}^k))^2}$$

***ADAM [15]**

$$\hat{\mathbf{H}}_k = \beta \hat{\mathbf{H}}_{k-1} + (1-\beta)\mathrm{diag}(\nabla f(\mathbf{x}^k))^2$$
$$\mathbf{H}_k = \sqrt{\hat{\mathbf{H}}_k / (1-\beta^k)}$$

AdaGrad - Adaptive gradient method with $\mathbf{H}_k = \lambda_k \mathbf{I}$

- If $\mathbf{H}_k = \lambda_k \mathbf{I}$, it becomes gradient descent method with adaptive step-size $\frac{\alpha_k}{\lambda_k}$.

How step-size adapts?

If gradient $\|\nabla f(\mathbf{x}^k)\|$ is large/small $\rightarrow$ AdaGrad adjusts step-size α_k/λ_k smaller/larger

Adaptive gradient descent (AdaGrad with $\mathbf{H}_k = \lambda_k \mathbf{I}$) [16]

1. Set $Q^0 = 0$.
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} Q^k &= Q^{k-1} + \|\nabla f(\mathbf{x}^k)\|^2 \\ \mathbf{H}_k &= \sqrt{Q^k} I \\ \mathbf{x}^{k+1} &= \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \nabla f(\mathbf{x}^k) \end{cases}$$

AdaGrad - Adaptive gradient method with $\mathbf{H}_k = \lambda_k \mathbf{I}$

- If $\mathbf{H}_k = \lambda_k \mathbf{I}$, it becomes gradient descent method with adaptive step-size $\frac{\alpha_k}{\lambda_k}$.

How step-size adapts?

If gradient $\|\nabla f(\mathbf{x}^k)\|$ is large/small $\rightarrow$ AdaGrad adjusts step-size α_k/λ_k smaller/larger

Adaptive gradient descent (AdaGrad with $\mathbf{H}_k = \lambda_k \mathbf{I}$) [16]

1. Set $Q^0 = 0$.
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} Q^k & = Q^{k-1} + \|\nabla f(\mathbf{x}^k)\|^2 \\ \mathbf{H}_k & = \sqrt{Q^k} I \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \nabla f(\mathbf{x}^k) \end{cases}$$

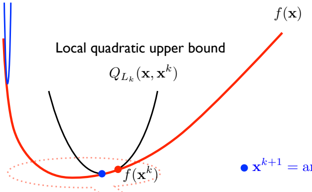
Adaptation through first-order information

- When $H_k = \lambda_k I$, AdaGrad estimates local geometry through gradient norms.
- Akin to estimating a local quadratic upper bound (majorization / minimization) using gradient history.

AdaGrad - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

Adaptation strategy with a positive diagonal matrix $\mathbf{D}_k$

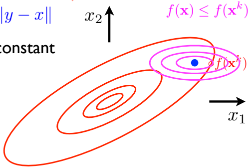
Adaptive step-size + coordinate-wise extension = adaptive step-size for each coordinate



- $\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L_k}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$

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

L is a global worst-case constant



$f(\mathbf{x}) \leq f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) + \frac{1}{2}\|\mathbf{x} - \mathbf{x}^k\|_{D_k^{-1}}^2$

applies only locally

AdaGrad - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

- Suppose $\mathbf{H}_k$ is diagonal,

$$\mathbf{H}_k := \begin{bmatrix} \lambda_{k,1} & & 0 \\ & \ddots & \\ 0 & & \lambda_{k,d} \end{bmatrix},$$

- For each coordinate i, we have different step-size $\frac{\alpha_k}{\lambda_{k,i}}$ is the step-size.

Adaptive gradient descent(AdaGrad with $\mathbf{H}_k = \mathbf{D}_k$)
1. Set $\mathbf{Q}^0$ =0. **2.** For $k = 0, 1, \ldots$, iterate $\begin{cases} \mathbf{Q}^k & = \mathbf{Q}^{k-1} + \mathrm{diag}(\nabla f(\mathbf{x}^k))^2 \\ \mathbf{H}_k & = \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \nabla f(\mathbf{x}^k) \end{cases}$

AdaGrad - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

◦ Suppose $\mathbf{H}_k$ is diagonal,

$$\mathbf{H}_k := \begin{bmatrix} \lambda_{k,1} & & 0 \\ & \ddots & \\ 0 & & \lambda_{k,d} \end{bmatrix},$$

◦ For each coordinate i, we have different step-size $\frac{\alpha_k}{\lambda_{k,i}}$ is the step-size.

Adaptive gradient descent(AdaGrad with $\mathbf{H}_k = \mathbf{D}_k$)

1. Set $\mathbf{Q}^0 = 0$.
2. For $k = 0, 1, \ldots$, iterate

$$\begin{cases} \mathbf{Q}^k & = \mathbf{Q}^{k-1} + \mathrm{diag}(\nabla f(\mathbf{x}^k))^2 \\ \mathbf{H}_k & = \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \nabla f(\mathbf{x}^k) \end{cases}$$

Adaptation across each coordinate

- When $\mathbf{H}_k = \mathbf{D}_k$, we adapt across each coordinate individually.
- Essentially, we have a finer treatment of the function we want to optimize.

Convergence rate for AdaGrad

Original convergence for a different function class

Consider a proper, convex function f such that it is G-Lipschitz continuous (NOT L-smooth). Let $D = \max_k \|\mathbf{x}^k - \mathbf{x}^\star\|_2$ and $\alpha_k = \frac{D}{\sqrt{2}}$. Define $\bar{\mathbf{x}}^k = (\sum_{i=1}^k \mathbf{x}^i)/k$. Then,

$$f(\bar{\mathbf{x}}^k) - f(\mathbf{x}^\star) \leq \frac{1}{k}\sqrt{2D^2 \sum_{i=1}^k \|\nabla f(\mathbf{x}^i)\|_2^2} \leq \frac{\sqrt{2}DG}{\sqrt{k}}$$

A more familiar convergence result [16]

Assume f is L-smooth, $D = \max_t \|\mathbf{x}^k - \mathbf{x}^\star\|_2$ and $\alpha_k = \frac{D}{\sqrt{2}}$. Define $\bar{\mathbf{x}}^k = (\sum_{i=1}^k \mathbf{x}^i)/k$. Then,

$$f(\bar{\mathbf{x}}^k) - f(\mathbf{x}^\star) \leq \frac{1}{k}\sqrt{2D^2 \sum_{i=1}^k \|\nabla f(\mathbf{x}^i)\|_2^2} \leq \frac{4D^2L}{k}$$

AcceleGrad - Adaptive gradient + Accelerated gradient [17]

Motivation behind AcceleGrad

Is it possible to achieve acceleration for when f is L-smooth, without knowing the Lipschitz constant?

- The answer is yes! See advanced material (AcceleGrad) at the end.
- A rough comparison of the accelerated methods:

Accelerated Gradient algorithm

1. Choose $\mathbf{x}^0 = \mathbf{y}^0 \in \mathrm{dom}\,(f)$
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} \mathbf{x}^{k+1} & = \mathbf{y}^k - \alpha \nabla f(\mathbf{y}^k) \\ \mathbf{y}^{k+1} & = \mathbf{x}^{k+1} + \gamma_{k+1}(\mathbf{x}^{k+1} - \mathbf{x}^k) \end{cases}$$

for some proper choice of α and γ_{k+1}.

AcceleGrad (Accelerated Adaptive Gradient Method)

1. Set $\mathbf{y}^0 = \mathbf{z}^0 = \mathbf{x}^0$
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} \tau_k & := 1/\alpha_k \\ \mathbf{x}^{k+1} & = \tau_k \mathbf{z}^k + (1 - \tau_k)\mathbf{y}^k \\ \mathbf{z}^{k+1} & = \mathbf{z}^k - \alpha_k \eta_k \nabla f(\mathbf{x}^k) \\ \mathbf{y}^{k+1} & = \mathbf{x}^{k+1} - \eta_k \nabla f(\mathbf{x}^k) \end{cases}$$

for $\alpha_k = (k+1)/4$ and

$$\eta_k = \frac{2D}{\sqrt{G^2 + \sum_{i=0}^{k} (\alpha_k)^2 \|\nabla f(\mathbf{x}^k)\|^2}}.$$

Performance of optimization algorithms

Time-to-reach ϵ

`time-to-reach` ϵ `= number of iterations to reach` ϵ $\times$ `per iteration time`

The **speed** of numerical solutions depends on two factors:

- **Convergence rate** determines the number of iterations needed to obtain an ϵ-optimal solution.
- **Per-iteration time** depends on the information oracles, implementation, and the computational platform.

In general, convergence rate and per-iteration time are inversely proportional.
Finding the **fastest** algorithm is tricky!

Performance of optimization algorithms (convex)

A non-exhaustive comparison:

Assumptions on f	Algorithm	Convergence rate	Iteration complexity
L-smooth	Gradient descent	Sublinear $(1/k)$	One gradient
	AdaGrad	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	AcceleGrad	Sublinear $(1/k^2)$	One gradient
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
L-smooth and μ-strongly convex	Gradient descent	Linear (e^{-k})	One gradient
	Accelerated GD	Linear (e^{-k})	One gradient
	Newton method	Linear (e^{-k}), Quadratic	One gradient, one linear system

Gradient descent:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha \nabla f(\mathbf{x}^k),$$

where the stepsize is chosen appropriately, $\alpha \in (0, \frac{2}{L})$

AdaGrad:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha^k \nabla f(\mathbf{x}^k),$$

where scalar version of the step size is
$\alpha^k = \frac{D}{\sqrt{\sum_{i=1}^k \|\nabla f(x^i)\|^2}}$

Performance of optimization algorithms (convex)

A non-exhaustive comparison:

Assumptions on f	Algorithm	Convergence rate	Iteration complexity
L-smooth	Gradient descent	Sublinear $(1/k)$	One gradient
	AdaGrad	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	AcceleGrad	Sublinear $(1/k^2)$	One gradient
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
L-smooth and μ-strongly convex	Gradient descent	Linear (e^{-k})	One gradient
	Accelerated GD	Linear (e^{-k})	One gradient
	Newton method	Linear (e^{-k}), Quadratic	One gradient, one linear system

Accelerated gradient descent:

$$\mathbf{x}^{k+1} = \mathbf{y}^k - \alpha \nabla f(\mathbf{y}^k)$$
$$\mathbf{y}^{k+1} = \mathbf{x}^{k+1} + \gamma_{k+1}(\mathbf{x}^{k+1} - \mathbf{x}^k).$$

for some proper choice of α and γ_{k+1}.

AcceleGrad:

$$\mathbf{x}^{k+1} = \tau_k \mathbf{z}^k + (1-\tau_k)\mathbf{y}^k$$
$$\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha_k \eta_k \nabla f(\mathbf{x}^k)$$
$$\mathbf{y}^{k+1} = \mathbf{x}^{k+1} - \eta_k \nabla f(\mathbf{x}^k).$$

for $\alpha_k = (k+1)/4$, $\tau_k = 1/\alpha_k$ and

$$\eta_k = \frac{2D}{\sqrt{G^2 + \sum_{i=0}^{k} (\alpha_k)^2 \|\nabla f(\mathbf{x}^k)\|^2}}.$$

Performance of optimization algorithms (convex)

A non-exhaustive comparison:

Assumptions on f	Algorithm	Convergence rate	Iteration complexity
L-smooth	Gradient descent	Sublinear $(1/k)$	One gradient
	AdaGrad	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	AcceleGrad	Sublinear $(1/k^2)$	One gradient
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
L-smooth and μ-strongly convex	Gradient descent	Linear (e^{-k})	One gradient
	Accelerated GD	Linear (e^{-k})	One gradient
	Newton method	Linear (e^{-k}), Quadratic	One gradient, one linear system

The main computation of the Newton method requires the solution of the linear system

$$\nabla^2 f(\mathbf{x}^k)\mathbf{p}^k = -\nabla f(\mathbf{x}^k) \ .$$

The gradient method for non-convex optimization

Remarks:
- ◦ Gradient descent does not match lower bounds in convex setting.
- ◦ How about non-convex problems?

Lower bounds for non-convex problems [5]

Assume f is L-gradient Lipschitz and non-convex. Then any first-order method must satisfy,

$$\|\nabla f(\mathbf{x}^k)\|^2 = \Omega\left(\frac{1}{k}\right)$$

Observations:
- ◦ Gradient descent is optimal for non-convex problems, up to some constant factor!
- ◦ Acceleration for non-convex, L-Lipschitz gradient functions is not as meaningful.

Recall: Gradient descent

Problem (Unconstrained optimization problem)

Consider the following minimization problem:

$$f^\star = \min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x})$$

$f(\mathbf{x})$ *is proper and closed.*

Gradient descent

Choose a starting point $\mathbf{x}^0$ and iterate

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k)$$

where α_k is a step-size to be chosen so that $\mathbf{x}^k$ converges to $\mathbf{x}^\star$.

	f is L-smooth & convex	f is L-gradient Lipschitz & non-convex
GD	$O(1/k)$ (fast)	$O(1/k)$ (optimal)
AGD	$O(1/k^2)$ (optimal)	$O(1/k)$ (optimal) [13]

Recall: Gradient descent

Problem (Unconstrained optimization problem)

Consider the following minimization problem:

$$f^\star = \min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x})$$

$f(\mathbf{x})$ *is proper and closed.*

Gradient descent

Choose a starting point $\mathbf{x}^0$ and iterate

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k)$$

where α_k is a step-size to be chosen so that $\mathbf{x}^k$ converges to $\mathbf{x}^\star$.

	f is L-smooth & **convex**	f is L-gradient Lipschitz & **non-convex**
GD	$O(1/k)$ (fast)	$O(1/k)$ (optimal)
AGD	$O(1/k^2)$ (optimal)	$O(1/k)$ (optimal) [13]

Why should we study anything else?

Statistical learning with streaming data

- Recall that statistical learning seeks to find a $h^\star \in \mathcal{H}$ that minimizes the *expected* risk,

$$h^\star \in \arg\min_{h \in \mathcal{H}} \left\{ R(h) := \mathbb{E}_{(\mathbf{a},b)} \left[L(h(\mathbf{a}), b) \right] \right\}.$$

Abstract gradient method

$$h^{k+1} = h^k - \alpha_k \nabla R(h^k) = h^k - \alpha_k \mathbb{E}_{(\mathbf{a},b)}[\nabla L(h^k(\mathbf{a}), b)].$$

This can not be implemented in practice as the distribution of (a, b) is unknown.

Statistical learning with streaming data

◦ Recall that statistical learning seeks to find a $h^\star \in \mathcal{H}$ that minimizes the *expected* risk,

$$h^\star \in \arg\min_{h\in\mathcal{H}} \left\{ R(h) := \mathbb{E}_{(\mathbf{a},b)} \left[L(h(\mathbf{a}), b) \right] \right\}.$$

Abstract gradient method

$$h^{k+1} = h^k - \alpha_k \nabla R(h^k) = h^k - \alpha_k \mathbb{E}_{(\mathbf{a},b)}[\nabla L(h^k(\mathbf{a}), b)].$$

This can not be implemented in practice as the distribution of (a, b) is unknown.

◦ In practice, data can arrive in a *streaming* way.

A parametric example: Markowitz portfolio optimization

$$\mathbf{x}^\star := \min_{\mathbf{x}\in\mathcal{X}} \left\{ \mathbb{E}\left[|b - \langle \mathbf{x}, \mathbf{a} \rangle|^2 \right] \right\}$$

- $h_{\mathbf{x}}(\cdot) = \langle \mathbf{x}, \cdot \rangle$
- $b \in \mathbb{R}$ is the desired return & $\mathbf{a} \in \mathbb{R}^p$ are the stock returns
- $\mathcal{X}$ is intersection of the standard simplex and the constraint: $\langle \mathbf{x}, \mathbb{E}[\mathbf{a}] \rangle \geq \rho$.

Stochastic programming

Problem (Mathematical formulation)

Consider the following convex minimization problem:

$$f^\star = \min_{\mathbf{x}\in\mathbb{R}^p} \left\{ f(\mathbf{x}) := \mathbb{E}[f(\mathbf{x},\theta)] \right\}$$

- θ *is a random vector whose probability distribution is supported on set* Θ.
- $f(\mathbf{x}) := \mathbb{E}[f(\mathbf{x},\theta)]$ *is proper, closed, and convex.*
- *The solution set* $\mathcal{S}^\star := \{\mathbf{x}^\star \in \mathrm{dom}\,(f) : f(\mathbf{x}^\star) = f^\star\}$ *is nonempty.*

Stochastic gradient descent (SGD)

Stochastic gradient descent (SGD)

1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ and $(\alpha_k)_{k\in\mathbb{N}} \in]0, +\infty[^{\mathbb{N}}$.
2. For $k = 0, 1, \ldots$ perform:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k G(\mathbf{x}^k, \theta_k).$$

- $G(\mathbf{x}^k, \theta_k)$ is an unbiased estimate of the full gradient:

$$\mathbb{E}[G(\mathbf{x}^k, \theta_k)] = \nabla f(\mathbf{x}^k).$$

Stochastic gradient descent (SGD)

Stochastic gradient descent (SGD)

1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ and $(\alpha_k)_{k\in\mathbb{N}} \in]0, +\infty[^{\mathbb{N}}$.
2. For $k = 0, 1, \ldots$ perform:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k G(\mathbf{x}^k, \theta_k).$$

◦ $G(\mathbf{x}^k, \theta_k)$ is an unbiased estimate of the full gradient:

$$\mathbb{E}[G(\mathbf{x}^k, \theta_k)] = \nabla f(\mathbf{x}^k).$$

Remarks:

◦ The cost of computing $G(\mathbf{x}^k, \theta_k)$ is n times cheaper than that of $\nabla f(\mathbf{x}^k)$.

◦ As $G(\mathbf{x}^k, \theta_k)$ is an unbiased estimate of the full gradient, SGD would perform well.

◦ We assume $\{\theta_k\}$ are jointly independent.

◦ SGD is not a monotonic descent method.

Example: Convex optimization with finite sums

Convex optimization with finite sums

The problem

$$\arg\min_{\mathbf{x}\in\mathbb{R}^p} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{j=1}^{n} f_j(\mathbf{x}) \right\},$$

can be rewritten as

$$\arg\min_{\mathbf{x}\in\mathbb{R}^p} \{ f(\mathbf{x}) := \mathbb{E}_i[f_i(\mathbf{x})] \}, \qquad i \text{ is uniformly distributed over } \{1, 2, \cdots, n\}.$$

A stochastic gradient descent (SGD) variant for finite sums

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \nabla f_i(\mathbf{x}^k) \qquad i \text{ is uniformly distributed over} \{1, ..., n\}$$

Remarks:

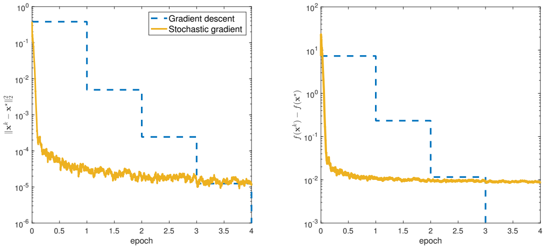
- ∘ Note: $\mathbb{E}_i[\nabla f_i(\mathbf{x}^k)] = \sum_{j=1}^{n} \nabla f_j(\mathbf{x}^k)/n = \nabla f(\mathbf{x}^k)$.
- ∘ The computational cost of SGD per iteration is p.

Synthetic least-squares problem

$$\min_{\mathbf{x}} \left\{ f(\mathbf{x}) := \frac{1}{2n}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 : \mathbf{x}\in\mathbb{R}^p \right\}$$

Setup

- $\mathbf{A} := \mathrm{randn}(n,p)$ - standard Gaussian $\mathcal{N}(0,\mathbb{I})$, with $n = 10^4$, $p = 10^2$.
- $\mathbf{x}^\natural$ is 50 sparse with zero mean Gaussian i.i.d. entries, normalized to $\|\mathbf{x}^\natural\|_2 = 1$.
- $\mathbf{b} := \mathbf{A}\mathbf{x}^\natural + \mathbf{w}$, where $\mathbf{w}$ is Gaussian white noise with variance 1.



○ 1 epoch $= 1$ pass over the full gradient

Convergence of SGD when the objective is not strongly convex

Theorem (decaying step-size [25])

Assume

- $\mathbb{E}[\|\mathbf{x}^k - \mathbf{x}^\star\|^2] \leq D^2$ *for all* k,
- $\mathbb{E}[\|G(\mathbf{x}^k, \theta_k)\|^2] \leq M^2$ *(bounded gradient)*,
- $\alpha_k = \alpha_0 / \sqrt{k}$.

Then

$$\mathbb{E}[f(\mathbf{x}^k) - f(\mathbf{x}^\star)] \leq \left(\frac{D^2}{\alpha_0} + \alpha_0 M^2 \right) \frac{2 + \log k}{\sqrt{k}}.$$

Observation: ∘ $\mathcal{O}(1/\sqrt{k})$ rate is optimal for SGD if we do not consider the strong convexity.

Convergence of SGD for strongly convex problems I

Theorem (strongly convex objective, fixed step-size [4])

Assume

- f *is* μ*-strongly convex and* L*-smooth,*
- $\mathbb{E}[\|G(\mathbf{x}^k, \theta_k)\|^2]_2 \leq \sigma^2 + M\|\nabla f(\mathbf{x}^k)\|_2^2$ *(bounded variance),*
- $\alpha_k = \alpha \leq \frac{1}{LM}$.

Then

$$\mathbb{E}[f(\mathbf{x}^k) - f(\mathbf{x}^\star)] \leq \frac{\alpha L \sigma^2}{2\mu} + (1 - \mu\alpha)^{k-1}\left(f(\mathbf{x}^1) - f^\star\right).$$

Observations:
- ∘ Converge fast (linearly) to a neighborhood around $\mathbf{x}^\star$
- ∘ Zero variance ($\sigma = 0$) $\implies$ linear convergence
- ∘ Smaller step-sizes $\alpha \implies$ converge to a better point, but with a slower rate

Convergence of SGD for strongly convex problems II

Theorem (strongly convex objective, decaying step-size [4])

Assume

- *f is μ-strongly convex and L-smooth,*
- $\mathbb{E}[\|G(\mathbf{x}^k, \theta_k)\|^2]_2 \leq \sigma^2 + M\|\nabla f(\mathbf{x}^k)\|_2^2$ *(bounded variance),*
- $\alpha_k = \frac{c}{k_0+k}$ *with some appropriate constants c and k_0.*

Then

$$\mathbb{E}[\|\mathbf{x}^k - \mathbf{x}^\star\|^2] \leq \frac{C}{k+1},$$

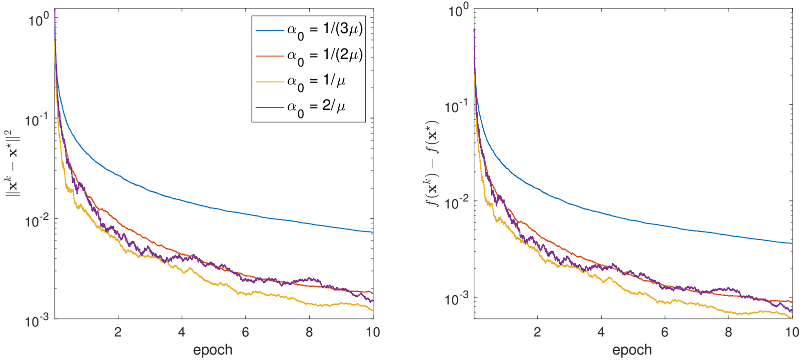
where C is a constant independent of k.

Observations: ∘ Using the smooth property,

$$\mathbb{E}[f(\mathbf{x}^k) - f(\mathbf{x}^\star)] \leq L\mathbb{E}[\|\mathbf{x}^k - \mathbf{x}^\star\|^2] \leq \frac{C}{k+1}.$$

∘ The rate is optimal if $\sigma^2 > 0$ with the assumption of strongly-convexity.

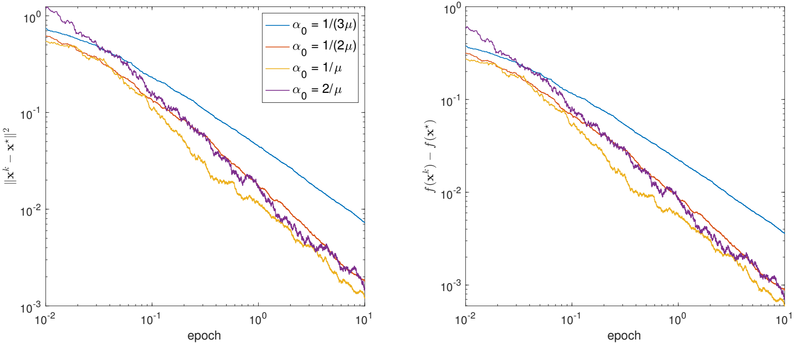
Example: SGD with different step sizes



Setup

- ◦ Synthetic least-squares problem as before
- ◦ $\alpha_k = \alpha_0/(k + k_0)$.

Example: SGD with different step sizes



Setup

- Synthetic least-squares problem as before
- $\alpha_k = \alpha_0/(k + k_0)$.

Observation: ◦ $\alpha_0 = 1/\mu$ is the best choice.

Comparison with GD

$$\boxed{f^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{j=1}^{n} f_j(\mathbf{x}) \right\}.}$$

∘ f: μ-strongly convex with L-Lipschitz smooth.

	rate	iteration complexity	cost per iteration	total cost
GD	ρ^k	$\log(1/\epsilon)$	n	$n\log(1/\epsilon)$
SGD	$1/k$	$1/\epsilon$	1	$1/\epsilon$

Remark: ∘ SGD is more favorable when n is large — large-scale optimization problems

Motivation for SGD with Averaging

- SGD iterates tend to oscillate around global minimizers
- Averaging iterates can reduce the oscillation effect
- Two types of averaging:

$$\bar{\mathbf{x}}^k = \frac{1}{k}\sum_{j=1}^{k} \alpha_j \mathbf{x}^j \quad \text{(vanilla averaging)}$$

$$\bar{\mathbf{x}}^k = \frac{\sum_{j=1}^{k} \alpha_j \mathbf{x}^j}{\sum_{j=1}^{k} \alpha_j} \quad \text{(weighted averaging)}$$

Convergence for SGD-A I: non-strongly convex case

Stochastic gradient method with averaging (SGD-A)
1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ and $(\alpha_k)_{k\in\mathbb{N}} \in]0, +\infty[^{\mathbb{N}}$. **2a.** For $k = 0, 1, \ldots$ perform: $\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k G(\mathbf{x}^k, \theta_k)$. **2b.** $\bar{\mathbf{x}}^k = (\sum_{j=0}^k \alpha_j)^{-1} \sum_{j=0}^k \alpha_j \mathbf{x}^j$.

Theorem (Convergence of SGD-A [19])

Let $D = \|\mathbf{x}^0 - \mathbf{x}^\star\|$ *and* $\mathbb{E}[\|G(\mathbf{x}^k, \theta_k)\|^2] \leq M^2$.
Then,

$$\mathbb{E}[f(\bar{\mathbf{x}}^{k+1}) - f(\mathbf{x}^\star)] \leq \frac{D^2 + M^2 \sum_{j=0}^k \alpha_j^2}{2\sum_{j=0}^k \alpha_j}.$$

In addition, choosing $\alpha_k = D/(M\sqrt{k+1})$, *we get,*

$$\mathbb{E}[f(\bar{\mathbf{x}}^k) - f(\mathbf{x}^\star)] \leq \frac{MD(2 + \log k)}{\sqrt{k}}.$$

Observation: ◦ Same convergence rate with vanilla SGD.

Convergence for SGD-A II: strongly convex case

Stochastic gradient method with averaging (SGD-A)

1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ and $(\alpha_k)_{k\in\mathbb{N}} \in]0, +\infty[^{\mathbb{N}}$.
2a. For $k = 0, 1, \ldots$ perform:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k G(\mathbf{x}^k, \theta_k).$$

2b. $\bar{\mathbf{x}}^k = \frac{1}{k}\sum_{j=1}^{k} \mathbf{x}^j$.

Theorem (Convergence of SGD-A [24])

Assume

- *f is μ-strongly convex,*
- $\mathbb{E}[\|G(\mathbf{x}^k, \theta_k)\|^2] \leq M^2$,
- $\alpha_k = \alpha_0/k$ *for some* $\alpha_0 \geq 1/\mu$.

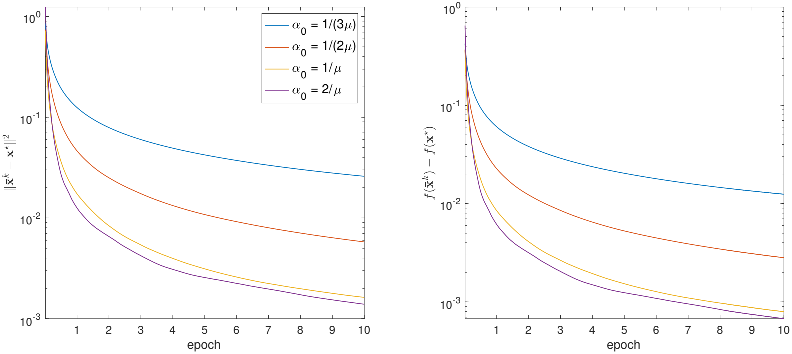
Then

$$\mathbb{E}[f(\bar{\mathbf{x}}^k) - f(\mathbf{x}^\star)] \leq \frac{\alpha_0 M^2(1 + \log k)}{2k}.$$

Observation: ◦ Same convergence rate with vanilla SGD.

Example: SGD-A method with different step sizes

$$\min_{\mathbf{x}} \left\{ f(\mathbf{x}) := \frac{1}{2n} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 : \mathbf{x} \in \mathbb{R}^p \right\}$$

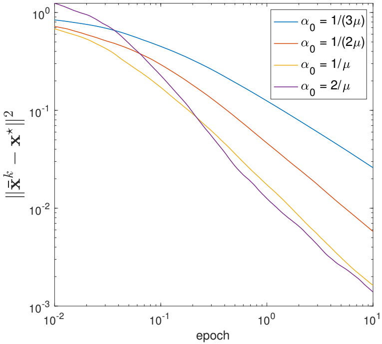


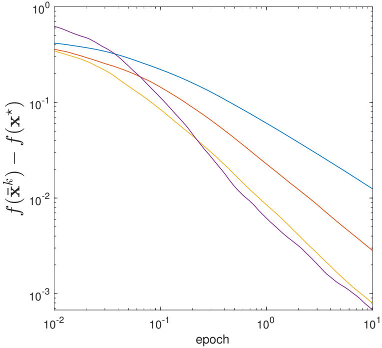
Setup

- ∘ Synthetic least-squares problem as before
- ∘ $\alpha_k = \alpha_0/(k + k_0)$.

Example: SGD-A method with different step sizes

$$\min_{\mathbf{x}} \left\{ f(\mathbf{x}) := \frac{1}{2n}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 : \mathbf{x} \in \mathbb{R}^p \right\}$$





Setup

- Synthetic least-squares problem as before
- $\alpha_k = \alpha_0/(k + k_0)$.

Observations:

- SGD-A is more stable than SGD.
- $\alpha_0 = 2/\mu$ is the best choice.

Least mean squares algorithm

Least-square regression problem

Solve

$$\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathbb{R}^p} \left\{ f(\mathbf{x}) := \frac{1}{2}\mathbb{E}_{(\mathbf{a},b)}(\langle \mathbf{a}, \mathbf{x}\rangle - b)^2 \right\},$$

given i.i.d. samples $\{(\mathbf{a}_j, b_j)\}_{j=1}^n$ (particularly in a streaming way).

Stochastic gradient method with averaging

1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ and $\alpha > 0$.
2a. For $k = 1, \ldots, n$ perform:

$$\mathbf{x}^k = \mathbf{x}^{k-1} - \alpha \left(\langle \mathbf{a}_k, \mathbf{x}^{k-1}\rangle - b_k \right) \mathbf{a}_k.$$

2b. $\bar{\mathbf{x}}^k = \frac{1}{k+1}\sum_{j=0}^{k} \mathbf{x}^j$.

$O(1/n)$ convergence rate, without strongly convexity [3]

Let $\|\mathbf{a}_j\|_2 \leq R$ and $|\langle \mathbf{a}_j, \mathbf{x}^\star\rangle - b_j| \leq \sigma$ a.s.. Pick $\alpha = 1/(4R^2)$. Then

$$\mathbb{E} f(\bar{\mathbf{x}}^{n-1}) - f^* \leq \frac{2}{n}\left(\sigma\sqrt{p} + R\|\mathbf{x}^0 - \mathbf{x}^\star\|_2\right)^2.$$

Popular SGD Variants

∘ Mini-batch SGD: For each iteration,

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \frac{1}{b} \sum_{\theta \in \Gamma} G(\mathbf{x}^k, \theta).$$

- α_k: step-size
- b : mini-batch size
- Γ : a set of random variables θ of size b

∘ Accelerated SGD (Nesterov accelerated technique)

∘ SGD with Momentum

∘ Adaptive stochastic methods: AdaGrad...

SGD - Non-convex stochastic optimization

- SGD is not as well-studied for non-convex problems as for convex problems.
- There is a gap between SGD's practical performance and theoretical understanding.
- Recall SGD update rule:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k G(\mathbf{x}^k, \theta)$$

Theorem (A well-known result for SGD & Non-convex problems [12])

Let f be a non-convex and L-smooth function. Set $\alpha_k = \min\left\{\frac{1}{L}, \frac{C}{\sigma\sqrt{T}}\right\}$, $\forall k = 1, ..., T$, where σ^2 is the variance of the gradients and $C > 0$ is constant. Then,

$$\mathbb{E}[\|\nabla f(\mathbf{x}^R)\|^2] = O\left(\frac{\sigma}{\sqrt{T}}\right),$$

where $\mathbb{P}(R = k) = \frac{2\alpha_k - L\alpha_k^2}{\sum_{k=1}^{T}(2\alpha_k - L\alpha_k^2)}$.

Lower bounds in non-convex optimization

Assumptions on f	Additional assumptions	Sample complexity
L-smooth	Deterministic Oracle $f(\mathbf{x}^0) - \inf_{\mathbf{x}} f(\mathbf{x}) \leq \Delta$	$\Omega(\Delta L \epsilon^{-2})$[6]
L_1-smooth L_2-Lipschitz Hessian	Deterministic Oracle $f(\mathbf{x}^0) - \inf_{\mathbf{x}} f(\mathbf{x}) \leq \Delta$	$\Omega(\Delta L_1^{3/7} L_2^{2/7} \epsilon^{-12/7})$[6]
L-smooth	$\mathbb{E}[G(\mathbf{x}, \theta)] = \nabla f(x)$ $\mathbb{E}[\\|G(\mathbf{x}, \theta) - \nabla f(\mathbf{x})\\|^2] \leq \sigma^2$ $f(\mathbf{x}^0) - \inf_{\mathbf{x}} f(\mathbf{x}) \leq \Delta$	$\Omega(\Delta L \sigma^2 \epsilon^{-4})$[2]
$G(\mathbf{x}, \theta)$ has averaged L-Lipschitz gradient $\implies$ L-smooth	$\mathbb{E}[G(\mathbf{x}, \theta)] = \nabla f(x)$ $\mathbb{E}[\\|G(\mathbf{x}, \theta) - \nabla f(\mathbf{x})\\|^2] \leq \sigma^2$ $f(\mathbf{x}^0) - \inf_{\mathbf{x}} f(\mathbf{x}) \leq \Delta$	$\Omega(\Delta L \sigma \epsilon^{-3} + \sigma^2 \epsilon^{-2})$[2]
$f(\mathbf{x}) := \frac{1}{n} \sum_{i=1}^n f_i(\mathbf{x})$ $f_i(\mathbf{x})$ has averaged L-Lipschitz gradient $\implies$ L-smooth	Access to $\nabla f_i(\mathbf{x})$ $f(\mathbf{x}^0) - \inf_{\mathbf{x}} f(\mathbf{x}) \leq \Delta$ $n \leq O(\epsilon^{-4})$[1]	$\Omega(\Delta L \sqrt{n} \epsilon^{-2})$[10]

- Measure of stationarity: $\|\nabla f(\mathbf{x})\| \leq \epsilon$ or $\mathbb{E}[\|\nabla f(\mathbf{x})\| \leq \epsilon$
- Sample complexity: $\#$ of total oracle calls (deterministic or stochastic gradients)
- Averaged L-Lipschitz gradient: $\mathbb{E}\left[\|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})\|^2\right] \leq L^2 \|\mathbf{x} - \mathbf{y}\|^2$
- $G(\mathbf{x}, \theta)$ denotes a stochastic gradient estimate for f at $\mathbf{x}$ with randomness governed by θ.

[1] We have $n \leq O(\epsilon^{-4})$ in order to match the respective *upper bound* of $O(n + \sqrt{n}\epsilon^{-2})$ achieved by [10]

Wrap up!

- The remaining slides in this lecture are advanced material.
- Lecture on Monday!

*Enhancements

Two enhancements

1. Line-search for estimating L for both GD and AGD.
2. Restart strategies for AGD.

⋆Enhancements

Two enhancements

1. Line-search for estimating L for both GD and AGD.
2. Restart strategies for AGD.

When do we need a line-search procedure?

We can use a line-search procedure for both GD and AGD when

- L is **known** but it is expensive to evaluate;
- The global constant L usually does not capture the local behavior of f or it is **unknown**.

*Enhancements

Two enhancements

1. Line-search for estimating L for both GD and AGD.
2. Restart strategies for AGD.

When do we need a line-search procedure?

We can use a line-search procedure for both GD and AGD when

- L is **known** but it is expensive to evaluate;
- The global constant L usually does not capture the local behavior of f or it is **unknown**.

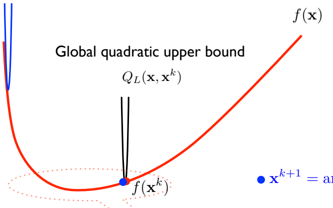
Line-search

At each iteration, we try to find a constant L_k that satisfies:

$$f(\mathbf{x}^{k+1}) \leq Q_{L_k}(\mathbf{x}^{k+1}, \mathbf{y}^k) := f(\mathbf{y}^k) + \langle \nabla f(\mathbf{y}^k), \mathbf{x}^{k+1} - \mathbf{y}^k \rangle + \frac{L_k}{2} \|\mathbf{x}^{k+1} - \mathbf{y}^k\|_2^2.$$

Here: $L_0 > 0$ is given (e.g., $L_0 := c\frac{\|\nabla f(\mathbf{x}^1) - \nabla f(\mathbf{x}^0)\|_2}{\|\mathbf{x}^1 - \mathbf{x}^0\|_2}$) for $c \in (0, 1]$.

⋆How can we better adapt to the local geometry?

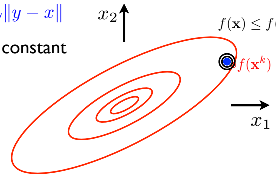


- $\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L}{2} \|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$

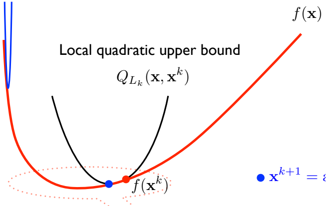
$\|\nabla f(x) - \nabla f(y)\| \leq L \|y - x\|$

L is a global worst-case constant

$f(\mathbf{x}) \leq f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T (\mathbf{x} - \mathbf{x}^k) + \frac{L}{2} \|\mathbf{x} - \mathbf{x}^k\|_2^2$



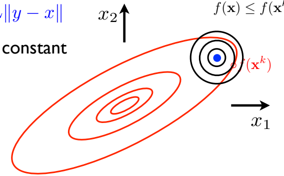
⋆How can we better adapt to the local geometry?



● $\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L_k}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$

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

L is a global worst-case constant



$f(\mathbf{x}) \leq f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) + \frac{L_k}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2$

applies only locally

*Enhancements

Why do we need a restart strategy?

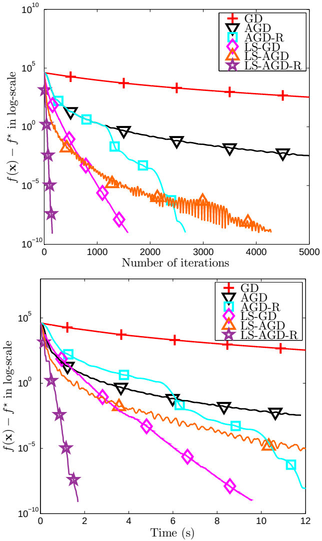
- AGD-μL requires knowledge of μ and AGD-L does not have optimal convergence for strongly convex f.
- AGD is non-monotonic (i.e., $f(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k)$ is not always satisfied).
- AGD has a periodic behavior, where the momentum depends on the local condition number $\kappa = L/\mu$.
- A **restart strategy** tries to reset this momentum whenever we observe high periodic behavior. We often use function values but other strategies are possible.

Restart strategies

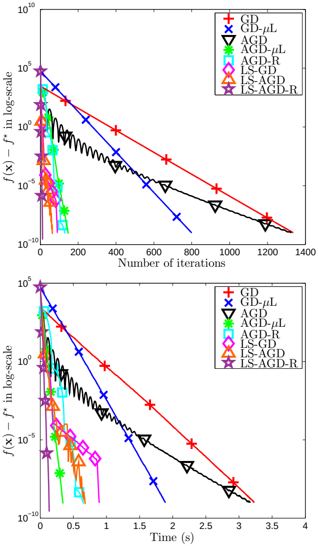
1. **O'Donoghue - Candes's strategy [22]:** There are at least three options: Restart with fixed number of iterations, restart based on objective values, and restart based on a gradient condition.
2. **Giselsson-Boyd's strategy [14]:** Do not require $t_k = 1$ and do not necessary require function evaluations.
3. **Fercoq-Qu's strategy [11]:** Unconditional periodic restart for strongly convex functions. Do not require the strong convexity parameter.

⋆Example: Ridge regression

Case 1: $n = 500, p = 2000, \rho = 0$



Case 2: $n = 500, p = 2000, \rho = 0.01\lambda_p(\mathbf{A}^T\mathbf{A})$



*AcceleGrad - Adaptive gradient + Accelerated gradient [17]

Motivation behind AcceleGrad

Is it possible to achieve acceleration when f is L-smooth, without knowing the Lipschitz constant?

AcceleGrad (Accelerated Adaptive Gradient Method)

Input : $\mathbf{x}^0 \in \mathcal{K}$, diameter D, weights $\{\alpha_k\}_{k\in\mathbb{N}}$, learning rate $\{\eta_k\}_{k\in\mathbb{N}}$

1. Set $\mathbf{y}^0 = \mathbf{z}^0 = \mathbf{x}^0$
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} \tau_k & := 1/\alpha_k \\ \mathbf{x}^{k+1} & = \tau_k \mathbf{z}^k + (1-\tau_k)\mathbf{y}^k, \text{define } \mathbf{g}_k := \nabla f(\mathbf{x}^{k+1}) \\ \mathbf{z}^{k+1} & = \Pi_{\mathcal{K}}(\mathbf{z}^k - \alpha_k \eta_k \mathbf{g}_k) \\ \mathbf{y}^{k+1} & = \mathbf{x}^{k+1} - \eta_k \mathbf{g}_k \end{cases}$$

Output : $\overline{\mathbf{y}}^k \propto \sum_{i=0}^{k-1} \alpha_i \mathbf{y}^{i+1}$

where $\Pi_{\mathcal{K}}(\mathbf{y}) = \arg\min_{\mathbf{x}\in\mathcal{K}} \langle \mathbf{x}-\mathbf{y}, \mathbf{x}-\mathbf{y}\rangle$ (projection onto $\mathcal{K}$).

Remark: ◦ This is essentially the **MD + GD** scheme [1], with an adaptive step size!

*AcceleGrad - Properties and convergence

Learning rate and weight computation

Assume that function f has uniformly bounded gradient norms $\|\nabla f(\mathbf{x}^k)\|^2 \leq G^2$, i.e., f is G-Lipschitz continuous. AcceleGrad uses the following weights and learning rate:

$$\alpha_k = \frac{k+1}{4}, \quad \eta_k = \frac{2D}{\sqrt{G^2 + \sum_{\tau=0}^{k} \alpha_\tau^2 \|\nabla f(\mathbf{x}_{\tau+1})\|^2}}$$

◦ Similar to RmsProp, AcceleGrad assignes greater weights to recent gradients.

Convergence rate of AcceleGrad

Assume that f is convex and L-smooth. Let K be a convex set with bounded diameter D, and assume $\mathbf{x}^\star \in K$. Define $\bar{\mathbf{y}}^k = (\sum_{i=0}^{k-1} \alpha_i \mathbf{y}^{i+1}) / (\sum_{i=0}^{k-1} \alpha_i)$. Then,

$$f(\bar{\mathbf{y}}^k) - \min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) \leq O\left(\frac{DG + LD^2 \log(LD/G)}{k^2}\right)$$

If f is only convex and G-Lipschitz, then

$$f(\bar{\mathbf{y}}^k) - \min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) \leq O\left(GD\sqrt{\log k}/\sqrt{k}\right)$$

*Example: Logistic regression

Problem (Logistic regression)

Given $\mathbf{A} \in \{0,1\}^{n\times p}$ *and* $\mathbf{b} \in \{-1,+1\}^n$, *solve:*

$$f^\star := \min_{\mathbf{x},\beta}\left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{j=1}^{n} \log\left(1+\exp\left(-\mathbf{b}_j(\mathbf{a}_j^T\mathbf{x}+\beta)\right)\right)\right\}.$$

Real data

- Real data: a4a with $\mathbf{A} \in \mathbb{R}^{n\times d}$, where $n = 4781$ data points, $d = 122$ features
- All methods are run for $T = 10000$ iterations

⋆RMSProp - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

What could be improved over AdaGrad?

1. Gradients have equal weights in step size.
2. Consider a *steep* function, flat around minimum $\rightarrow$ slow convergence at flat region.

*RMSProp - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

What could be improved over AdaGrad?

1. Gradients have equal weights in step size.
2. Consider a *steep* function, flat around minimum $\rightarrow$ slow convergence at flat region.

AdaGrad with $\mathbf{H}_k = \mathbf{D}_k$	RMSProp
1. Set $\mathbf{Q}_0 = 0$. **2.** For $k = 0, 1, \ldots,$ iterate $\begin{cases} \mathbf{Q}^k &= \mathbf{Q}^{k-1} + \mathrm{diag}(\nabla f(\mathbf{x}^k))^2 \\ \mathbf{H}_k &= \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} &= \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \nabla f(\mathbf{x}^k) \end{cases}$	**1.** Set $\mathbf{Q}_0 = 0$. **2.** For $k = 0, 1, \ldots,$ iterate $\begin{cases} \mathbf{Q}^k &= \beta \mathbf{Q}^{k-1} + (1-\beta)\mathrm{diag}(\nabla f(\mathbf{x}^k))^2 \\ \mathbf{H}_k &= \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} &= \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \nabla f(\mathbf{x}^k) \end{cases}$

*RMSProp - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

What could be improved over AdaGrad?

1. Gradients have equal weights in step size.
2. Consider a *steep* function, flat around minimum $\rightarrow$ slow convergence at flat region.

AdaGrad with $\mathbf{H}_k = \mathbf{D}_k$

1. Set $\mathbf{Q}_0 = 0$.
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} \mathbf{Q}^k & = \mathbf{Q}^{k-1} + \mathrm{diag}(\nabla f(\mathbf{x}^k))^2 \\ \mathbf{H}_k & = \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \nabla f(\mathbf{x}^k) \end{cases}$$

RMSProp

1. Set $\mathbf{Q}_0 = 0$.
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} \mathbf{Q}^k & = \beta \mathbf{Q}^{k-1} + (1-\beta)\mathrm{diag}(\nabla f(\mathbf{x}^k))^2 \\ \mathbf{H}_k & = \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \nabla f(\mathbf{x}^k) \end{cases}$$

- RMSProp uses weighted averaging with constant β
- Recent gradients have greater importance

*ADAM - Adaptive moment estimation

Over-simplified idea of ADAM

$$\text{RMSProp} + \text{2nd order moment estimation} = \text{ADAM}$$

*ADAM - Adaptive moment estimation

Over-simplified idea of ADAM

RMSProp + 2nd order moment estimation = ADAM

ADAM
Input. Step size α, exponential decay rates $\beta_1, \beta_2 \in [0, 1)$
1. Set $\mathbf{m}_0, \mathbf{v}_0 = 0$ **2.** For $k = 0, 1, \ldots,$ iterate $\left\{\begin{array}{ll} \mathbf{g}_k & = \nabla f(\mathbf{x}^{k-1}) \\ \mathbf{m}_k & = \beta_1 \mathbf{m}_{k-1} + (1-\beta_1)\mathbf{g}_k \leftarrow \text{1st order estimate} \\ \mathbf{v}_k & = \beta_2 \mathbf{v}_{k-1} + (1-\beta_2)\mathbf{g}_k^2 \leftarrow \text{2nd order estimate} \\ \hat{\mathbf{m}}_k & = \mathbf{m}_k/(1-\beta_1^k) \leftarrow \text{Bias correction} \\ \hat{\mathbf{v}}_k & = \mathbf{v}_k/(1-\beta_2^k) \leftarrow \text{Bias correction} \\ \mathbf{H}_k & = \sqrt{\hat{\mathbf{v}}_k} + \epsilon \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha \hat{\mathbf{m}}_k / \mathbf{H}_k \end{array}\right.$
Output : $\mathbf{x}^k$

(Every vector operation is an element-wise operation)

*Non-convergence of ADAM and a new method: AmsGrad

- It has been shown that ADAM may not converge for *some* objective functions [23].
- An ADAM alternative is proposed that is proved to be convergent [23].

AmsGrad
Input. Step size $\{\alpha_k\}_{k\in\mathbb{N}}$, exponential decay rates $\{\beta_{1,k}\}_{k\in\mathbb{N}}$, $\beta_2 \in [0,1)$
1. Set $\mathbf{m}_0 = 0, \mathbf{v}_0 = 0$ and $\hat{\mathbf{v}}_0 = 0$ **2.** For $k = 1, 2, \ldots,$ iterate $\left\{\begin{array}{ll} \mathbf{g}_k & = G(\mathbf{x}^k, \theta) \\ \mathbf{m}_k & = \beta_{1,k}\mathbf{m}_{k-1} + (1-\beta_{1,k})\mathbf{g}_k \leftarrow \text{1st order estimate} \\ \mathbf{v}_k & = \beta_2\mathbf{v}_{k-1} + (1-\beta_2)\mathbf{g}_k^2 \leftarrow \text{2nd order estimate} \\ \hat{\mathbf{v}}_k & = \max\{\hat{\mathbf{v}}_{k-1}, \mathbf{v}_k\} \text{ and } \hat{\mathbf{V}}_k = \text{diag}(\hat{\mathbf{v}}_k) \\ \mathbf{H}_k & = \sqrt{\hat{\mathbf{v}}_k} \\ \mathbf{x}^{k+1} & = \Pi_{\mathcal{X}}^{\sqrt{\hat{\mathbf{V}}_k}}(\mathbf{x}^k - \alpha_k\hat{\mathbf{m}}_k/\mathbf{H}_k) \end{array}\right.$
Output : $\mathbf{x}^k$

where $\Pi_{\mathcal{K}}^{\mathbf{A}}(\mathbf{y}) = \arg\min_{\mathbf{x}\in\mathcal{K}} \langle(\mathbf{x}-\mathbf{y}), \mathbf{A}(\mathbf{x}-\mathbf{y})\rangle$ (weighted projection onto $\mathcal{K}$).
(Every vector operation is an element-wise operation)

*AdaGrad & AmsGrad for non-convex optimization

Theorem (AdaGrad convergence rate: stochastic, non-convex [28])

Assume f is non-convex and L-smooth, such that $\|\nabla f(\mathbf{x})\|^2 \leq G^2$ and $f^\star = \inf_{\mathbf{x}} f(\mathbf{x}) > \infty$. Also consider bounded variance for unbiased gradient estimates, i.e., $\mathbb{E}\left[\|G(\mathbf{x},\theta) - \nabla f(\mathbf{x})\|^2 | \mathbf{x}\right] \leq \sigma^2$. Then with probability $1-\delta$,

$$\min_{i \in \{1,..,k-1\}} \|\nabla f(\mathbf{x}^i)\|^2 = \tilde{O}\left(\frac{\sigma}{\delta^{3/2}\sqrt{k}}\right)$$

∘ **Note:** As $1-\delta \to 1$, the rate deteriorates by a factor of $\delta^{-3/2}$.

Theorem (AmsGrad convergence rate 1: stochastic, non-convex [7])

Let $\mathbf{g}_k = G(x^k, \theta)$. Assume $|\mathbf{g}_{1,i}| > c > 0$, $\forall i \in [d]$ and $\|\mathbf{g}_k\| \leq G$. Consider a non-increasing sequence $\beta_{1,k}$ and $\beta_{1,k} \leq \beta_1 \in [0,1)$. Set $\alpha_k = 1/\sqrt{k}$. Then,

$$\min_{i \in \{1,..,k-1\}} \mathbb{E}\left[\|\nabla f(\mathbf{x}^i)\|^2\right] = O\left(\frac{\log k}{\sqrt{k}}\right).$$

*AdaGrad & AmsGrad for non-convex optimization

Theorem (AdaGrad convergence rate: stochastic, non-convex [28])

Assume f is non-convex and L-smooth, such that $\|\nabla f(\mathbf{x})\|^2 \leq G^2$ and $f^\star = \inf_{\mathbf{x}} f(\mathbf{x}) > \infty$. Also consider bounded variance for unbiased gradient estimates, i.e., $\mathbb{E}\left[\|G(\mathbf{x},\theta) - \nabla f(\mathbf{x})\|^2 | \mathbf{x}\right] \leq \sigma^2$. Then with probability $1-\delta$,

$$\min_{i \in \{1,..,k-1\}} \|\nabla f(\mathbf{x}^i)\|^2 = \tilde{O}\left(\frac{\sigma}{\delta^{3/2}\sqrt{k}}\right)$$

- **Note:** As $1-\delta \to 1$, the rate deteriorates by a factor of $\delta^{-3/2}$.

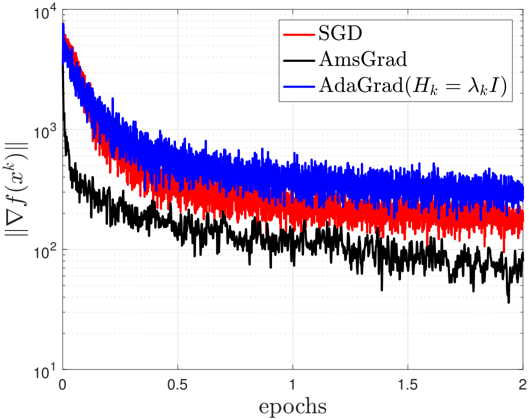
Theorem (AmsGrad convergence rate 2: stochastic, non-convex [29])

Consider $f : \mathbb{R}^d \to \mathbb{R}$ to be non-convex ans L-smooth. Assume $\|G(\mathbf{x},\theta)\|_\infty \leq G_\infty$ and set $\alpha_k = 1/\sqrt{dT}$. Also define $\mathbf{x}_{out} = \mathbf{x}^k$, for $k = 1,\dots,T$ with probability $\alpha^k / \sum_{i=1}^T \alpha_i$. Then,

$$\mathbb{E}\left[\|\nabla f(\mathbf{x}_{out})\|^2\right] = O\left(\sqrt{\frac{d}{T}}\right).$$

*Example: Logistic regression with non-convex regularizer

- Synthetic data: $\mathbf{A} \in \mathbb{R}^{n\times d}$, $n = 2000$, $d = 200$.
- Batch size: 20 samples.
- Algorithms: SGD, AdaGrad, AmsGrad.



*Adaptive methods for stochastic optimization

Remark

- Adaptive methods have extensive applications in stochastic optimization.
- We will see **another nature** of adaptive methods in this lecture.
- Mild additional assumption: **bounded variance** of gradient estimates.

*AdaGrad for stochastic optimization

∘ Only modification: $\nabla f(\mathbf{x}) \Rightarrow G(\mathbf{x}, \theta)$

AdaGrad with $\mathbf{H}_k = \lambda_k \mathbf{I}$ [16]
1. Set Q^0 =0. **2.** For $k = 0, 1, \ldots,$ iterate $\begin{cases} Q^k & = Q^{k-1} + \lVert G(\mathbf{x}^k, \theta) \rVert^2 \\ \mathbf{H}_k & = \sqrt{Q^k}\mathbf{I} \\ \mathbf{x}^{k+1} & = \mathbf{x}_t - \alpha_k \mathbf{H}_k^{-1} G(\mathbf{x}^k, \theta) \end{cases}$

Theorem (Convergence rate: stochastic, convex optimization [16])

Assume f is convex and L-smooth, such that minimizer of f lies in a convex, compact set $\mathcal{K}$ with diameter D. Also consider bounded variance for unbiased gradient estimates, i.e., $\mathbb{E}\left[\lVert G(\mathbf{x}, \theta) - \nabla f(\mathbf{x}) \rVert^2 | \mathbf{x}\right] \leq \sigma^2$. Then,

$$\mathbb{E}[f(\mathbf{x}^k)] - \min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x}) = O\left(\frac{\sigma D}{\sqrt{k}}\right)$$

∘ AdaGrad is **adaptive** also in the sense that it adapts to nature of the oracle.

*AcceleGrad for stochastic optimization

- Similar to AdaGrad, replace $\nabla f(\mathbf{x}) \Rightarrow G(\mathbf{x}, \theta)$

AcceleGrad (Accelerated Adaptive Gradient Method)
Input : $\mathbf{x}^0 \in \mathcal{K}$, diameter D, weights $\{\alpha_k\}_{k\in\mathbb{N}}$, learning rate $\{\eta_k\}_{k\in\mathbb{N}}$
1. Set $\mathbf{y}^0 = \mathbf{z}^0 = \mathbf{x}^0$ **2.** For $k = 0, 1, \ldots,$ iterate $\left\{\begin{array}{ll} \tau_k & := 1/\alpha_k \\ \mathbf{x}^{k+1} & = \tau_t \mathbf{z}^k + (1-\tau_k)\mathbf{y}^k, \text{define } \mathbf{g}_k := \nabla f(\mathbf{x}^{k+1}) \\ \mathbf{z}^{k+1} & = \Pi_{\mathcal{K}}(\mathbf{z}^k - \alpha_k \eta_k \mathbf{g}_k) \\ \mathbf{y}^{k+1} & = \mathbf{x}^{k+1} - \eta_k \mathbf{g}_k \end{array}\right.$
Output : $\overline{\mathbf{y}}^k \propto \sum_{i=0}^{k-1} \alpha_i \mathbf{y}^{i+1}$

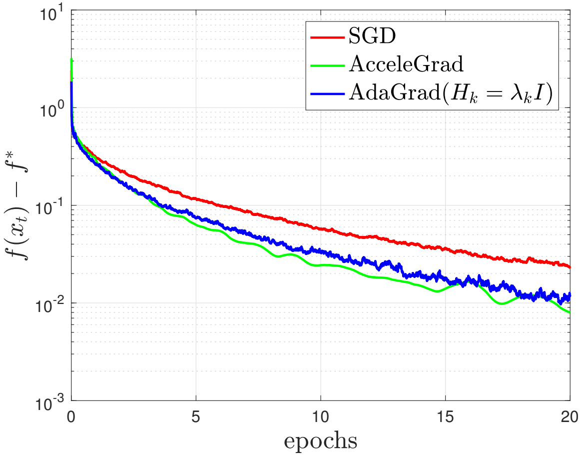
Theorem (Convergence rate [17])

Assume f is convex and G-Lipschitz and that minimizer of f lies in a convex, compact set $\mathcal{K}$ with diameter D. Also consider bounded variance for unbiased gradient estimates, i.e., $\mathbb{E}\left[\|G(\mathbf{x},\theta) - \nabla f(\mathbf{x})\|^2 | \mathbf{x}\right] \leq \sigma^2$. Then,

$$\mathbb{E}[f(\overline{\mathbf{y}}^k)] - \min_{\mathbf{x}} f(\mathbf{x}) = O\left(\frac{GD\sqrt{\log k}}{\sqrt{k}}\right).$$

*Example: Synthetic least squares

- $\mathbf{A} \in \mathbb{R}^{n \times d}$, where $n = 200$ and $d = 50$.
- Number of epochs: 20.
- Algorithms: SGD, AdaGrad & AcceleGrad.



*Newton method

- Fast (local) convergence but expensive per iteration cost
- **Useful** when warm-started near a solution

*Newton method

- Fast (local) convergence but expensive per iteration cost
- **Useful** when warm-started near a solution

Local quadratic approximation using the Hessian

- Obtain a local quadratic approximation using the second-order Taylor series approximation to $f(\mathbf{x}^k + \mathbf{p})$:

$$f(\mathbf{x}^k + \mathbf{p}) \approx f(\mathbf{x}^k) + \langle \mathbf{p}, \nabla f(\mathbf{x}^k) \rangle + \frac{1}{2} \langle \mathbf{p}, \nabla^2 f(\mathbf{x}^k) \mathbf{p} \rangle$$

*Newton method

- Fast (local) convergence but expensive per iteration cost
- **Useful** when warm-started near a solution

Local quadratic approximation using the Hessian

- Obtain a local quadratic approximation using the second-order Taylor series approximation to $f(\mathbf{x}^k + \mathbf{p})$:

$$f(\mathbf{x}^k + \mathbf{p}) \approx f(\mathbf{x}^k) + \langle \mathbf{p}, \nabla f(\mathbf{x}^k) \rangle + \frac{1}{2} \langle \mathbf{p}, \nabla^2 f(\mathbf{x}^k) \mathbf{p} \rangle$$

- The Newton direction is the vector $\mathbf{p}^k$ that minimizes $f(\mathbf{x}^k + \mathbf{p})$; assuming the Hessian $\nabla^2 f_k$ to be **positive definite**:

$$\nabla^2 f(\mathbf{x}^k)\mathbf{p}^k = -\nabla f(\mathbf{x}^k) \quad \Leftrightarrow \quad \mathbf{p}^k = -\left(\nabla^2 f(\mathbf{x}^k)\right)^{-1} \nabla f(\mathbf{x}^k)$$

⋆Newton method

- Fast (local) convergence but expensive per iteration cost
- **Useful** when warm-started near a solution

Local quadratic approximation using the Hessian

- Obtain a local quadratic approximation using the second-order Taylor series approximation to $f(\mathbf{x}^k + \mathbf{p})$:

$$f(\mathbf{x}^k + \mathbf{p}) \approx f(\mathbf{x}^k) + \langle \mathbf{p}, \nabla f(\mathbf{x}^k) \rangle + \frac{1}{2}\langle \mathbf{p}, \nabla^2 f(\mathbf{x}^k)\mathbf{p} \rangle$$

- The Newton direction is the vector $\mathbf{p}^k$ that minimizes $f(\mathbf{x}^k + \mathbf{p})$; assuming the Hessian $\nabla^2 f_k$ to be **positive definite**:

$$\nabla^2 f(\mathbf{x}^k)\mathbf{p}^k = -\nabla f(\mathbf{x}^k) \quad \Leftrightarrow \quad \mathbf{p}^k = -\left(\nabla^2 f(\mathbf{x}^k)\right)^{-1} \nabla f(\mathbf{x}^k)$$

- A unit step-size $\alpha_k = 1$ can be chosen near convergence:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \left(\nabla^2 f(\mathbf{x}^k)\right)^{-1} \nabla f(\mathbf{x}^k) \; .$$

*Newton method

- Fast (local) convergence but expensive per iteration cost
- **Useful** when warm-started near a solution

Local quadratic approximation using the Hessian

- Obtain a local quadratic approximation using the second-order Taylor series approximation to $f(\mathbf{x}^k + \mathbf{p})$:

$$f(\mathbf{x}^k + \mathbf{p}) \approx f(\mathbf{x}^k) + \langle \mathbf{p}, \nabla f(\mathbf{x}^k) \rangle + \frac{1}{2} \langle \mathbf{p}, \nabla^2 f(\mathbf{x}^k) \mathbf{p} \rangle$$

- The Newton direction is the vector $\mathbf{p}^k$ that minimizes $f(\mathbf{x}^k + \mathbf{p})$; assuming the Hessian $\nabla^2 f_k$ to be **positive definite**:

$$\nabla^2 f(\mathbf{x}^k) \mathbf{p}^k = -\nabla f(\mathbf{x}^k) \quad \Leftrightarrow \quad \mathbf{p}^k = -\left(\nabla^2 f(\mathbf{x}^k)\right)^{-1} \nabla f(\mathbf{x}^k)$$

- A unit step-size $\alpha_k = 1$ can be chosen near convergence:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \left(\nabla^2 f(\mathbf{x}^k)\right)^{-1} \nabla f(\mathbf{x}^k) \, .$$

Remark

- For $f \in \mathcal{F}_L^{2,1}$ but $f \notin \mathcal{F}_{L,\mu}^{2,1}$, the Hessian may not always be positive definite.

*(Local) Convergence of Newton method

Lemma

Assume f is a twice differentiable convex function with minimum at $\mathbf{x}^\star$ such that:

- $\nabla^2 f(\mathbf{x}^\star) \succeq \mu \mathbf{I}$ *for some* $\mu > 0$,
- $\|\nabla^2 f(\mathbf{x}) - \nabla^2 f(\mathbf{y})\|_{2\to 2} \leq M\|\mathbf{x} - \mathbf{y}\|_2$ *for some constant* $M > 0$ *and all* $\mathbf{x}, \mathbf{y} \in \mathrm{dom}(f)$.

Moreover, assume the starting point $\mathbf{x}^0 \in \mathrm{dom}(f)$ *is such that* $\|\mathbf{x}^0 - \mathbf{x}^\star\|_2 < \frac{2\mu}{3M}$.
Then, the Newton method iterates converge **quadratically***:*

$$\|\mathbf{x}^{k+1} - \mathbf{x}^\star\| \leq \frac{M\|\mathbf{x}^k - \mathbf{x}^\star\|_2^2}{2\left(\mu - M\|\mathbf{x}^k - \mathbf{x}^\star\|_2\right)}.$$

Remark

This is the fastest convergence rate we have seen so far, but it requires to solve a $p \times p$ linear system at each iteration, $\nabla^2 f(\mathbf{x}^k)\mathbf{p}^k = -\nabla f(\mathbf{x}^k)$!

*Locally quadratic convergence of the Newton method–I

Newton's method local quadratic convergence - Proof [21]

Since $\nabla f(\mathbf{x}^\star) = 0$ we have

$$\begin{aligned}\mathbf{x}^{k+1} - \mathbf{x}^\star &= \mathbf{x}^k - \mathbf{x}^\star - (\nabla^2 f(\mathbf{x}^k))^{-1}\nabla f(\mathbf{x}^k) \\ &= (\nabla^2 f(\mathbf{x}^k))^{-1}\left(\nabla^2 f(\mathbf{x}^k)(\mathbf{x}^k - \mathbf{x}^\star) - (\nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^\star))\right)\end{aligned}$$

By Taylor's theorem, we also have

$$\nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^\star) = \int_0^1 \nabla^2 f(\mathbf{x}^k + t(\mathbf{x}^\star - \mathbf{x}^k))(\mathbf{x}^k - \mathbf{x}^\star)dt$$

Combining the two above, we obtain

$$\begin{aligned}&\|\nabla^2 f(\mathbf{x}^k)(\mathbf{x}^k - \mathbf{x}^\star) - (\nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^\star))\| \\ &= \left\|\int_0^1 \left(\nabla^2 f(\mathbf{x}^k) - \nabla^2 f(\mathbf{x}^k + t(\mathbf{x}^\star - \mathbf{x}^k))\right)(\mathbf{x}^k - \mathbf{x}^\star)dt\right\| \\ &\leq \int_0^1 \left\|\nabla^2 f(\mathbf{x}^k) - \nabla^2 f(\mathbf{x}^k + t(\mathbf{x}^\star - \mathbf{x}^k))\right\| \|\mathbf{x}^k - \mathbf{x}^\star\|dt \\ &\leq M\|\mathbf{x}^k - \mathbf{x}^\star\|^2 \int_0^1 tdt = \frac{1}{2}M\|\mathbf{x}^k - \mathbf{x}^\star\|^2\end{aligned}$$

*Locally quadratic convergence of the Newton method–II

Newton's method local quadratic convergence - Proof [21].

- Recall

$$\mathbf{x}^{k+1} - \mathbf{x}^\star = (\nabla^2 f(\mathbf{x}^k))^{-1} \left(\nabla^2 f(\mathbf{x}^k)(\mathbf{x}^k - \mathbf{x}^\star) - (\nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^\star))\right)$$

$$\|\nabla^2 f(\mathbf{x}^k)(\mathbf{x}^k - \mathbf{x}^\star) - (\nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^\star))\| \leq \frac{1}{2} M \|\mathbf{x}^k - \mathbf{x}^\star\|^2$$

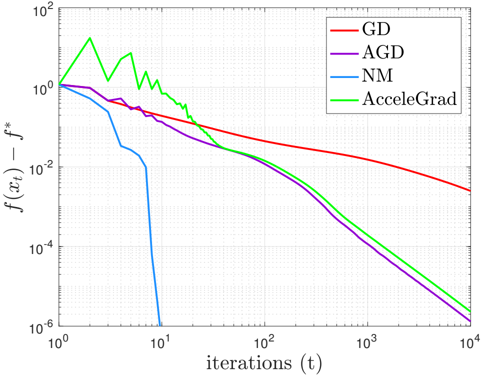
- Since $\nabla^2 f(\mathbf{x}^\star)$ is nonsingular, there must exist a radius r such that $\|(\nabla^2 f(\mathbf{x}^k))^{-1}\| \leq 2\|(\nabla^2 f(\mathbf{x}^\star))^{-1}\|$ for all $\mathbf{x}^k$ with $\|\mathbf{x}^k - \mathbf{x}^*\| \leq r$.
- Substituting, we obtain

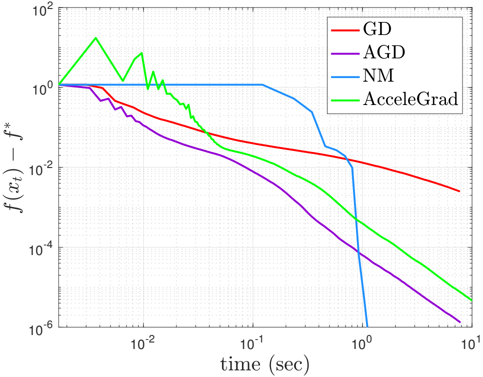
$$\|\mathbf{x}^{k+1} - \mathbf{x}^\star\| \leq M\|(\nabla^2 f(\mathbf{x}^\star))^{-1}\| \|\mathbf{x}^k - \mathbf{x}^\star\|^2 = \widetilde{M}\|\mathbf{x}^k - \mathbf{x}^\star\|^2,$$

  where $\widetilde{M} = M\|(\nabla^2 f(\mathbf{x}^\star))^{-1}\|$.
- If we choose $\|\mathbf{x}^0 - \mathbf{x}^\star\| \leq \min(r, 1/(2\widetilde{M}))$, we obtain by induction that the iterates $\mathbf{x}^k$ converge quadratically to $\mathbf{x}^\star$.

$\square$

*Example: Logistic regression - GD, AGD, AcceleGrad + NM





Parameters

- Newton's method: maximum number of iterations 30, tolerance 10^{-6}.
- For GD, AGD & AcceleGrad: maximum number of iterations 10000, tolerance 10^{-6}.
- Ground truth: Get a high accuracy approximation of $\mathbf{x}^\star$ and $f^\star$ by applying Newton's method for 200 iterations.

⋆*Approximating* Hessian: Quasi-Newton methods

Quasi-Newton methods use an approximate Hessian oracle and can be more scalable.

- Useful for $f(\mathbf{x}) := \sum_{i=1}^{n} f_i(\mathbf{x})$ with $n \gg p$.

Main ingredients

Quasi-Newton direction:

$$\mathbf{p}^k = -\mathbf{H}_k^{-1}\nabla f(\mathbf{x}^k) = -\mathbf{B}_k\nabla f(\mathbf{x}^k).$$

- Matrix $\mathbf{H}_k$, or its inverse $\mathbf{B}_k$, undergoes low-rank updates:
  - Rank 1 or 2 updates: famous Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm.
  - Limited memory BFGS (L-BFGS).
- Line-search: The step-size α_k is chosen to satisfy the **Wolfe conditions**:

$$f(\mathbf{x}^k + \alpha_k\mathbf{p}^k) \leq f(\mathbf{x}^k) + c_1\alpha_k\langle\nabla f(\mathbf{x}^k), \mathbf{p}^k\rangle \qquad \text{(sufficient decrease)}$$
$$\langle\nabla f(\mathbf{x}^k + \alpha_k\mathbf{p}^k), \mathbf{p}^k\rangle \geq c_2\langle\nabla f(\mathbf{x}^k), \mathbf{p}^k\rangle \qquad \text{(curvature condition)}$$

  with $0 < c_1 < c_2 < 1$. For quasi-Newton methods, we usually use $c_1 = 0.1$.
- Convergence is guaranteed under the Dennis & Moré condition [8].
- For more details on quasi-Newton methods, see Nocedal&Wright's book [21].

*Quasi-Newton methods

How do we update $\mathbf{B}_{k+1}$?

Suppose we have (note the coordinate change from $\mathbf{p}$ to $\bar{\mathbf{p}}$)

$$m_{k+1}(\bar{\mathbf{p}}) := f(\mathbf{x}^{k+1}) + \langle \nabla f(\mathbf{x}^{k+1}), \bar{\mathbf{p}} - \mathbf{x}^{k+1} \rangle + \frac{1}{2} \left\langle \mathbf{B}_{k+1}(\bar{\mathbf{p}} - \mathbf{x}^{k+1}), (\bar{\mathbf{p}} - \mathbf{x}^{k+1}) \right\rangle .$$

We require the gradient of m_{k+1} to match the gradient of f at $\mathbf{x}^k$ and $\mathbf{x}^{k+1}$.

- $\nabla m_{k+1}(\mathbf{x}^{k+1}) = \nabla f(\mathbf{x}^{k+1})$ as desired;
- For $\mathbf{x}^k$, we have
$$\nabla m_{k+1}(\mathbf{x}^k) = \nabla f(\mathbf{x}^{k+1}) + \mathbf{B}_{k+1}(\mathbf{x}^k - \mathbf{x}^{k+1})$$
which must be equal to $\nabla f(\mathbf{x}^k)$.
- Rearranging, we have that $\mathbf{B}_{k+1}$ must satisfy the **secant equation**
$$\mathbf{B}_{k+1}\mathbf{s}^k = \mathbf{y}^k$$
where $\mathbf{s}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$ and $\mathbf{y}^k = \nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k)$.
- The secant equation can be satisfied with a positive definite matrix $\mathbf{B}_{k+1}$ only if $\langle \mathbf{s}^k, \mathbf{y}^k \rangle > 0$, which is guaranteed to hold if the step-size α_k satisfies the Wolfe conditions.

*Quasi-Newton methods

BFGS method [21] (from Broyden, Fletcher, Goldfarb & Shanno)

The BFGS method arises from directly updating $\mathbf{H}_k = \mathbf{B}_k^{-1}$. The update on the inverse $\mathbf{B}$ is found by solving

$$\min_{\mathbf{H}} \|\mathbf{H} - \mathbf{H}_k\|_{\mathbf{W}} \quad \text{subject to } \mathbf{H} = \mathbf{H}^T \text{ and } \mathbf{H}\mathbf{y}^k = \mathbf{s}^k \tag{4}$$

The solution is a rank-2 update of the matrix $\mathbf{H}_k$:

$$\mathbf{H}_{k+1} = \mathbf{V}_k^T \mathbf{H}_k \mathbf{V}_k + \eta_k \mathbf{s}^k (\mathbf{s}^k)^T \ ,$$

where $\mathbf{V}_k = \mathbf{I} - \eta_k \mathbf{y}^k (\mathbf{s}^k)^T$.

- Initialization of $\mathbf{H}_0$ is an art. We can choose to set it to be an approximation of $\nabla^2 f(\mathbf{x}^0)$ obtained by finite differences or just a multiple of the identity matrix.

*Quasi-Newton methods

BFGS method [21] (from Broyden, Fletcher, Goldfarb & Shanno)

The BFGS method arises from directly updating $\mathbf{H}_k = \mathbf{B}_k^{-1}$. The update on the inverse $\mathbf{B}$ is found by solving

$$\min_{\mathbf{H}} \|\mathbf{H} - \mathbf{H}_k\|_{\mathbf{W}} \quad \text{subject to } \mathbf{H} = \mathbf{H}^T \text{ and } \mathbf{H}\mathbf{y}^k = \mathbf{s}^k \tag{4}$$

The solution is a rank-2 update of the matrix $\mathbf{H}_k$:

$$\mathbf{H}_{k+1} = \mathbf{V}_k^T \mathbf{H}_k \mathbf{V}_k + \eta_k \mathbf{s}^k (\mathbf{s}^k)^T \ ,$$

where $\mathbf{V}_k = \mathbf{I} - \eta_k \mathbf{y}^k (\mathbf{s}^k)^T$.

Theorem (Convergence of BFGS)

Let $f \in \mathcal{C}^2$. Assume that the BFGS sequence $\{\mathbf{x}^k\}$ converges to a point $\mathbf{x}^\star$ and $\sum_{k=1}^{\infty} \|\mathbf{x}^k - \mathbf{x}^\star\| \leq \infty$. Assume also that $\nabla^2 f(\mathbf{x})$ is Lipschitz continuous at $\mathbf{x}^\star$. Then $\mathbf{x}^k$ converges to $\mathbf{x}^\star$ at a **superlinear** *rate.*

Remarks

The proof shows that given the assumptions, the BFGS updates for $\mathbf{B}_k$ satisfy the Dennis & Moré condition, which in turn implies superlinear convergence.

*L-BFGS

Challenges for BFGS

- BFGS approach stores and applies a dense $p \times p$ matrix $\mathbf{H}_k$.
- When p is very large, $\mathbf{H}_k$ can prohibitively expensive to store and apply.

L(imited memory)-BFGS

- Do not store $\mathbf{H}_k$, but keep only the m most recent pairs $\{(\mathbf{s}^i, \mathbf{y}^i)\}$.
- Compute $\mathbf{H}_k \nabla f(\mathbf{x}_k)$ by performing a sequence of operations with $\mathbf{s}^i$ and $\mathbf{y}^i$:
  - Choose a temporary initial approximation $\mathbf{H}_k^0$.
  - Recursively apply $\mathbf{H}_{k+1} = \mathbf{V}_k^T \mathbf{H}_k \mathbf{V}_k + \eta_k \mathbf{s}^k (\mathbf{s}^k)^T$, m times starting from $\mathbf{H}_k^0$:

$$
\begin{aligned}
\mathbf{H}_k = & \left(\mathbf{V}_{k-1}^T \cdots \mathbf{V}_{k-m}^T\right) \mathbf{H}_k^0 \left(\mathbf{V}_{k-m} \cdots \mathbf{V}_{k-1}\right) \\
& + \eta_{k-m} \left(\mathbf{V}_{k-1}^T \cdots \mathbf{V}_{k-m+1}^T\right) \mathbf{s}^{k-m} (\mathbf{s}^{k-m})^T \left(\mathbf{V}_{k-m+1} \cdots \mathbf{V}_{k-1}\right) \\
& + \cdots \\
& + \eta_{k-1} \mathbf{s}^{k-1} (\mathbf{s}^{k-1})^T
\end{aligned}
$$

  - From the previous expression, we can compute $\mathbf{H}_k \nabla f(\mathbf{x}^k)$ recursively.
- Replace the oldest element in $\{\mathbf{s}^i, \mathbf{y}^i\}$ with $(\mathbf{s}^k, \mathbf{y}^k)$.
- From practical experience, $m \in (3, 50)$ does the trick.

*L-BFGS: A quasi-Newton method

Procedure for computing $\mathbf{H}_k \nabla f(\mathbf{x}^k)$
0. Recall $\eta_k = 1/\langle \mathbf{y}^k, \mathbf{s}^k \rangle$.
1. $\mathbf{q} = \nabla f(\mathbf{x}^k)$.
2. For $i = k-1, \ldots, k-m$ $$\begin{aligned} \alpha_i &= \eta_i \langle \mathbf{s}^i, \mathbf{q} \rangle \\ \mathbf{q} &= \mathbf{q} - \alpha_i \mathbf{y}^i. \end{aligned}$$
3. $\mathbf{r} = \mathbf{H}_k^0 \mathbf{q}$.
4. For $i = k-m, \ldots, k-1$ $$\begin{aligned} \beta &= \eta_i \langle \mathbf{y}^i, \mathbf{r} \rangle \\ \mathbf{r} &= \mathbf{r} + (\alpha_i - \beta) \mathbf{s}^i. \end{aligned}$$
5. $\mathbf{H}_k \nabla f(\mathbf{x}^k) = \mathbf{r}$.

Remarks

- Apart from the step $\mathbf{r} = \mathbf{H}_k^0 \mathbf{q}$, the algorithm requires only $4mp$ multiplications.
- If $\mathbf{H}_k^0$ is chosen to be diagonal, another p multiplications are needed.
- An effective initial choice is $\mathbf{H}_k^0 = \gamma_k \mathbf{I}$, where

$$\gamma_k = \frac{\langle \mathbf{s}^{k-1}, \mathbf{y}^{k-1} \rangle}{\langle \mathbf{y}^{k-1}, \mathbf{y}^{k-1} \rangle}$$

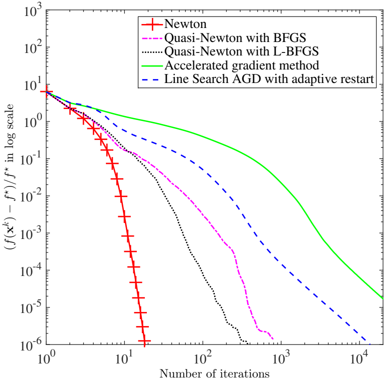
*L-BFGS: A quasi-Newton method

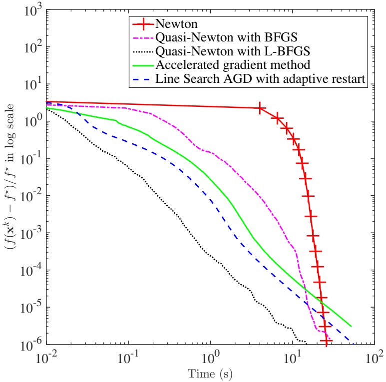
L-BFGS
1. Choose starting point $\mathbf{x}^0$ and $m > 0$. 2. For $k = 0, 1, \ldots$ 2.a Choose $\mathbf{H}_k^0$. 2.b Compute $\mathbf{p}^k = -\mathbf{H}_k \nabla f(\mathbf{x}^k)$ using the previous algorithm. 2.c Set $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}^k$, where α_k satisfies the Wolfe conditions. **if** $k > m$, discard the pair $\{\mathbf{s}^{k-m}, \mathbf{p}^{k-m}\}$ from storage. 2.d Compute and store $\mathbf{s}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$, $\mathbf{y}^k = \nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k)$.

Warning

L-BFGS updates does not guarantee positive semidefiniteness of the variable metric $\mathbf{H}_k$ in contrast to BFGS.

*Example: Logistic regression - numerical results





Parameters

- For BFGS, L-BFGS and Newton's method: maximum number of iterations 200, tolerance 10^{-6}. L-BFGS memory $m = 50$.
- For accelerated gradient method: maximum number of iterations 20000, tolerance 10^{-6}.
- Ground truth: Get a high accuracy approximation of $\mathbf{x}^\star$ and $f^\star$ by applying Newton's method for 200 iterations.

*Performance of optimization algorithms

Time-to-reach ϵ

`time-to-reach` ϵ `= number of iterations to reach` ϵ $\times$ `per iteration time`

The **speed** of numerical solutions depends on two factors:

- **Convergence rate** determines the number of iterations needed to obtain an ϵ-optimal solution.
- **Per-iteration time** depends on the information oracles, implementation, and the computational platform.

In general, convergence rate and per-iteration time are inversely proportional.
Finding the **fastest** algorithm is tricky! A non-exhaustive illustration:

Assumptions on f	Algorithm	Convergence rate	Iteration complexity
L-smooth	Gradient descent	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
L-smooth and μ-strongly convex	Gradient descent	Linear (e^{-k})	One gradient
	Accelerated GD	Linear (e^{-k})	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Linear (e^{-k}), Quadratic	One gradient, one linear system

*Performance of optimization algorithms

A non-exhaustive comparison:

Assumptions on f	Algorithm	Convergence rate	Iteration complexity
L-smooth	Gradient descent	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
L-smooth and μ-strongly convex	Gradient descent	Linear (e^{-k})	One gradient
	Accelerated GD	Linear (e^{-k})	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Linear (e^{-k}), Quadratic	One gradient, one linear system

Accelerated gradient descent:

$$\begin{aligned}\mathbf{x}^{k+1} &= \mathbf{y}^k - \alpha \nabla f(\mathbf{y}^k)\\ \mathbf{y}^{k+1} &= \mathbf{x}^{k+1} + \alpha_{k+1}(\mathbf{x}^{k+1} - \mathbf{x}^k).\end{aligned}$$

for some proper choice of α and α_{k+1}.

*Performance of optimization algorithms

A non-exhaustive comparison:

Assumptions on f	Algorithm	Convergence rate	Iteration complexity
L-smooth	Gradient descent	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
L-smooth and μ-strongly convex	Gradient descent	Linear (e^{-k})	One gradient
	Accelerated GD	Linear (e^{-k})	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Linear (e^{-k}), Quadratic	One gradient, one linear system

Main computations of the Quasi-Newton method is given by

$$\mathbf{p}^k = -\mathbf{B}_k^{-1}\nabla f(\mathbf{x}^k) \ ,$$

where $\mathbf{B}_k^{-1}$ is updated at each iteration by adding a rank-2 matrix.

*Performance of optimization algorithms

A non-exhaustive comparison:

Assumptions on f	Algorithm	Convergence rate	Iteration complexity
L-smooth	Gradient descent	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
L-smooth and μ-strongly convex	Gradient descent	Linear (e^{-k})	One gradient
	Accelerated GD	Linear (e^{-k})	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Linear (e^{-k}), Quadratic	One gradient, one linear system

The main computation of the Newton method requires the solution of the linear system

$$\nabla^2 f(\mathbf{x}^k)\mathbf{p}^k = -\nabla f(\mathbf{x}^k) \ .$$

*Randomized Kaczmarz algorithm

Problem

Given a full-column-rank matrix $\mathbf{A} \in \mathbb{R}^{n \times p}$ and $b \in \mathbb{R}^n$, solve the linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}.$$

Notations: $\mathbf{b} := (b_1, \ldots, b_n)^T$ and $\mathbf{a}_j^T$ is the j-th row of $\mathbf{A}$.

Randomized Kaczmarz algorithm (RKA)
1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$.
2. For $k = 0, 1, \ldots$ perform:
2a. Pick $j_k \in \{1, \cdots, n\}$ randomly with $\Pr(j_k = i) = \Vert\mathbf{a}_i\Vert_2^2 / \Vert\mathbf{A}\Vert_F^2$
2b. $\mathbf{x}^{k+1} = \mathbf{x}^k - \left(\langle \mathbf{a}_{j_k}, \mathbf{x}^k\rangle - b_{j_k}\right)\mathbf{a}_{j_k} / \Vert\mathbf{a}_{j_k}\Vert_2^2$.

Linear convergence [26]

Let $\mathbf{x}^\star$ be the solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$ and $\kappa = \Vert\mathbf{A}\Vert_F \Vert\mathbf{A}^{-1}\Vert$. Then

$$\mathbb{E}\Vert\mathbf{x}^k - \mathbf{x}^\star\Vert_2^2 \leq (1 - \kappa^{-2})^k \Vert\mathbf{x}^0 - \mathbf{x}^*\Vert_2^2$$

- RKA can be seen as a particular case of SGD [18].

References I

[1] Zeyuan Allen-Zhu and Lorenzo Orecchia.
Linear Coupling: An Ultimate Unification of Gradient and Mirror Descent.
In *Proceedings of the 8th Innovations in Theoretical Computer Science*, ITCS '17, 2017.
Full version available at `http://arxiv.org/abs/1407.1537`.

[2] Yossi Arjevani, Yair Carmon, John C. Duchi, Dylan J. Foster, Nathan Srebro, and Blake E. Woodworth.
Lower bounds for non-convex stochastic optimization.
ArXiv, abs/1912.02365, 2019.

[3] Francis Bach and Eric Moulines.
Non-strongly-convex smooth stochastic approximation with convergence rate o(1/n).
In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, pages 773–781, USA, 2013. Curran Associates Inc.

[4] Léon Bottou, Frank E. Curtis, and Jorge Nocedal.
Optimization methods for large-scale machine learning, 2016.
quantization overview.

[5] Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford.
Lower bounds for finding stationary points of non-convex , smooth high-dimensional functions.
2017.

References II

[6] Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford.
Lower bounds for finding stationary points II: first-order methods.
Math. Program., 185(1-2):315–355, 2021.

[7] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong.
On the convergence of a class of adam-type algorithms for non-convex optimization.
In *International Conference on Learning Representations*, 2019.

[8] JE Dennis and Jorge J Moré.
A characterization of superlinear convergence and its application to quasi-newton methods.
Mathematics of Computation, 28(126):549–560, 1974.

[9] 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.

[10] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang.
SPIDER: near-optimal non-convex optimization via stochastic path-integrated differential estimator.
In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 687–697, 2018.

References III

[11] Olivier Fercoq and Zheng Qu.
Restarting accelerated gradient methods with a rough strong convexity estimate.
2016.
arXiv:16009.07358v1.

[12] Saeed Ghadimi and Guanghui Lan.
Stochastic first-and zeroth-order methods for nonconvex stochastic programming.
SIAM Journal on Optimization, 23(4):2341–2368, 2013.

[13] Saeed Ghadimi and Guanghui Lan.
Accelerated gradient methods for nonconvex nonlinear and stochastic programming.
Math. Program., 156(1-2):59–99, 2016.

[14] Pontus Giselsson and Stephen Boyd.
Monotonicity and restart in fast gradient methods.
In *IEEE 53rd Ann. Conf. Decision and Control*, pages 5058–5063, 2014.

[15] Diederik Kingma and Jimmy Ba.
Adam: A method for stochastic optimization.
arXiv preprint arXiv:1412.6980, 2014.

References IV

[16] Kfir Levy.
Online to offline conversions, universality and adaptive minibatch sizes.
In *Advances in Neural Information Processing Systems*, pages 1613–1622, 2017.

[17] Kfir Levy, Alp Yurtsever, and Volkan Cevher.
Online adaptive methods, universality and acceleration.
In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018.

[18] Deanna Needell, Rachel Ward, and Nati Srebro.
Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm.
In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1017–1025. Curran Associates, Inc., 2014.

[19] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro.
Robust stochastic approximation approach to stochastic programming.
SIAM J. on Optimization, 19(4):1574–1609, January 2009.

[20] Yu. Nesterov.
Introductory Lectures on Convex Optimization: A Basic Course.
Kluwer, Boston, MA, 2004.

References V

[21] J. Nocedal and S.J. Wright.
Numerical Optimization.
Springer, 2006.

[22] Brendan O'Donoghue and Emmanuel Candes.
Adaptive restart for accelerated gradient schemes.
Found. Comput. Math., 15(3):715–732, 2015.

[23] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar.
On the convergence of adam and beyond.
In *International Conference on Learning Representations*, 2018.

[24] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter.
Pegasos: primal estimated sub-gradient solver for svm.
Mathematical Programming, 127(1):3–30, Mar 2011.

[25] Ohad Shamir and Tong Zhang.
Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes.
In *International Conference on Machine Learning*, pages 71–79, 2013.

References VI

[26] Thomas Strohmer and Roman Vershynin.
A randomized kaczmarz algorithm with exponential convergence.
Journal of Fourier Analysis and Applications, 15(2):262, Apr 2008.

[27] Tijmen Tieleman and Geoffrey Hinton.
Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.
COURSERA: Neural networks for machine learning, 4(2):26–31, 2012.

[28] Rachel Ward, Xiaoxia Wu, and Leon Bottou.
AdaGrad stepsizes: Sharp convergence over nonconvex landscapes.
In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6677–6686, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[29] Dongruo Zhou, Yiqi Tang, Ziyan Yang, Yuan Cao, and Quanquan Gu.
On the convergence of adaptive gradient methods for nonconvex optimization.
ArXiv, abs/1808.05671, 2018.