Mathematics of Data: From Theory to Computation

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 5: Optimality of Convergence rates. Accelerated Gradient/Tensor Descent Methods

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

EE-556 (Fall 2023)

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* $\mathcal{S}^\star := \{\mathbf{x}^\star \in \mathrm{dom}\,(f) : f(\mathbf{x}^\star) = f^\star\}$ *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

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

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 [23]

Question: ○ 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

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

Observations:
- ○ Gradient descent is $\mathcal{O}(1/k)$ for $\mathcal{F}_L^{\infty}$
- ○ It is also 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.
- The momentum $\mathbf{x}^{k+1} - \mathbf{x}^k$ acts like an "extra-gradient."

Global convergence of AGD [23]

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 [23]

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*

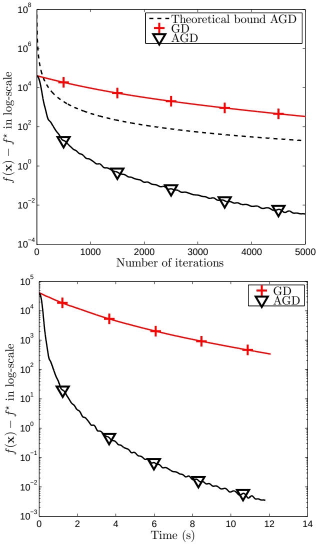
$$f(\mathbf{x}^k) - f^\star \leq L\left(1 - \sqrt{\frac{\mu}{L}}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2, \ \forall k \geq 0 \tag{2}$$

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

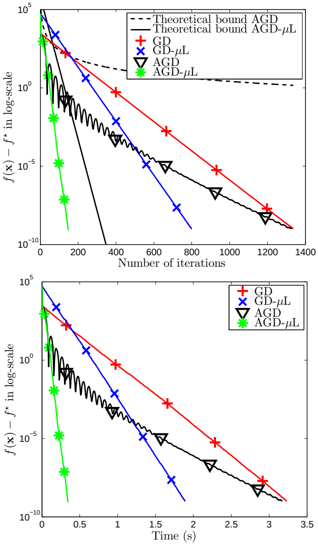
Observations:
- ∘ AGD-L's iterates are not guaranteed to converge in general.
- ∘ AGD-L does not have a **linear** convergence rate for L-smooth and μ-strongly convex.
- ∘ AGD-μL does, but needs to know μ.
- ∘ AGD achieves the iteration lowerbound within a constant!

Example: Ridge regression

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

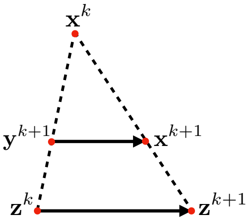


Case 2: $n = 500, p = 2000, \rho = 0.01\lambda_p(\mathbf{A}^T\mathbf{A})$



Hidden gems in AGD: The method of similar triangles

∘ There are several variants of Nesterov's AGD [26].



Accelerated Gradient Descent Algorithm

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

$$\begin{cases} t^{k+1} &= \frac{2}{k+1} \\ \mathbf{y}^{k+1} &= (1 - t^{k+1})\mathbf{x}^k + t^{k+1}\mathbf{z}^k \\ \mathbf{x}^{k+1} &= \mathbf{y}^{k+1} - \frac{1}{L}\nabla f(\mathbf{y}^{k+1}) \\ \mathbf{z}^{k+1} &= \mathbf{x}^{k+1} + \left(\frac{1}{t^{k+1}} - 1\right)(\mathbf{x}^{k+1} - \mathbf{x}^k) \end{cases}$$

Remarks:

∘ Triangles $(\mathbf{x}^k, \mathbf{y}^{k+1}, \mathbf{x}^{k+1})$ and $(\mathbf{x}^k, \mathbf{z}^k, \mathbf{z}^{k+1})$ are "similar."

∘ This geometric construction via averaging is typical of accelerated methods.

∘ Sequences $(\mathbf{y}^{k+1}, \mathbf{z}^{k+1})$ enable acceleration by estimating a lower-bound to the problem.

The extra-gradient algorithm

◦ Recall: The momentum-term $\mathbf{x}^{k+1} - \mathbf{x}^k$ in AGD acts like an "extra-gradient."

◦ However, the name extra-gradient is reserved for another algorithm approximating the proximal-point method:

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

Extra-gradient algorithm [13]

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

$$\begin{cases} \mathbf{x}^{k+1/2} & = \mathbf{x}^k - \gamma \nabla f(\mathbf{x}^k) \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \gamma \nabla f(\mathbf{x}^{k+1/2}) \end{cases}$$

- Pick $\gamma < \frac{1}{L}$.
- Define $\bar{\mathbf{x}}^{k+1/2} = \sum_{i=1}^{k} \mathbf{x}^{i+1/2}/k$
- $f(\bar{\mathbf{x}}^{k+1/2}) - f(\mathbf{x}^\star) \leq O\left(\frac{1}{k}\right)$

The extra-gradient algorithm

- Recall: The momentum-term $\mathbf{x}^{k+1} - \mathbf{x}^k$ in AGD acts like an "extra-gradient."
- However, the name extra-gradient is reserved for another algorithm approximating the proximal-point method:

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

Extra-gradient algorithm [13]
1. Choose $\mathbf{x}^0 \in \mathrm{dom}\,(f)$ **2.** For $k = 0, 1, \ldots,$ iterate $\begin{cases} \mathbf{x}^{k+1/2} & = \mathbf{x}^k - \gamma\nabla f(\mathbf{x}^k) \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \gamma\nabla f(\mathbf{x}^{k+1/2}) \end{cases}$

- Pick $\gamma < \frac{1}{L}$.
- Define $\bar{\mathbf{x}}^{k+1/2} = \sum_{i=1}^k \mathbf{x}^{i+1/2}/k$
- $f(\bar{\mathbf{x}}^{k+1/2}) - f(\mathbf{x}^\star) \leq O\left(\frac{1}{k}\right)$

Accelerated extra-gradient algorithm [7]
1. Set $\mathbf{x}^0 = \mathbf{z}^0 = \mathbf{x}^0$ **2.** For $k = 0, 1, \ldots,$ iterate $\begin{cases} \mathbf{x}^{k+1/2} & = \mathbf{x}^k - \alpha_k\gamma\nabla f(\tilde{\mathbf{x}}^k) \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k\gamma\nabla f(\bar{\mathbf{x}}^{k+1/2}) \end{cases}$

- Pick $\gamma < \frac{1}{L}$ and define $\alpha_k = O(k)$
- $\tilde{\mathbf{x}}^k = \frac{\alpha_k \mathbf{x}^k + \sum_{i=1}^{k-1} \alpha_i \mathbf{x}^{i+1/2}}{\sum_{i=1}^k \alpha_i}, \quad \bar{\mathbf{x}}^{k+1/2} = \frac{\sum_{i=1}^k \alpha_i \mathbf{x}^{i+1/2}}{\sum_{i=1}^k \alpha_i}$
- $f(\bar{\mathbf{x}}^{k+1/2}) - f(\mathbf{x}^\star) \leq O\left(\frac{1}{k^2}\right)$ [7]

Gradient descent vs. Accelerated gradient descent

Assumptions, step sizes and convergence rates

Gradient descent:

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

Accelerated Gradient Descent:

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

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.

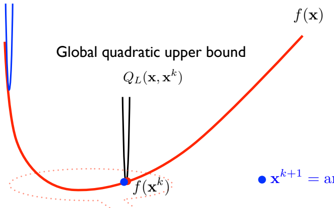
They do so by making use of the information from gradients and their norms.

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.

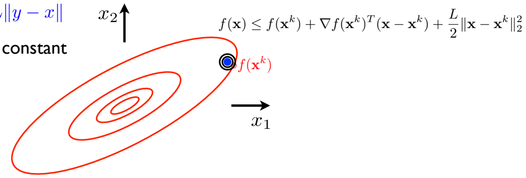
How can we better adapt to the local geometry?



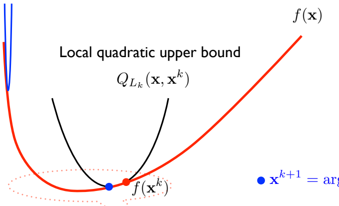
$$\bullet\ \mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$$

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

L is a global worst-case constant



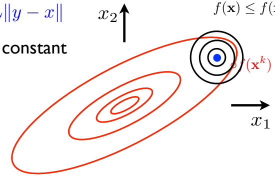
How can we better adapt to the local geometry?



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

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

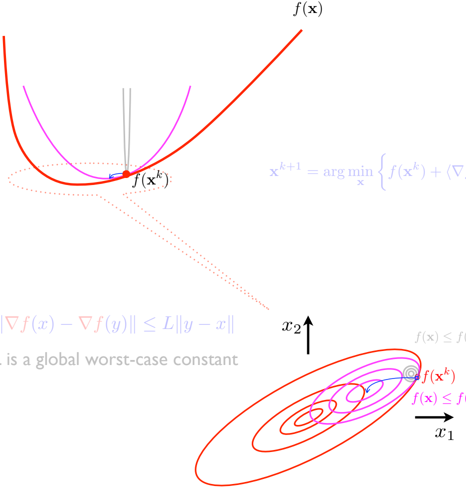
L is a global worst-case constant



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

applies only locally

How can we better adapt to the local geometry?



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

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

L is a global worst-case constant

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

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

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 (Scalar) [8, 19]

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

*RmsProp [33]

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

*ADAM [12]

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

AdaGrad - Adaptive gradient method with $\mathbf{H}_k = \lambda_k \mathbf{I}$ [19]

∘ 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 $\mathbf{H}_k = \lambda_k \mathbf{I}$) [15]

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 $\mathbf{H}_k = \lambda_k \mathbf{I}$ [19]

∘ 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 $\mathbf{H}_k = \lambda_k \mathbf{I}$) [15]

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

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

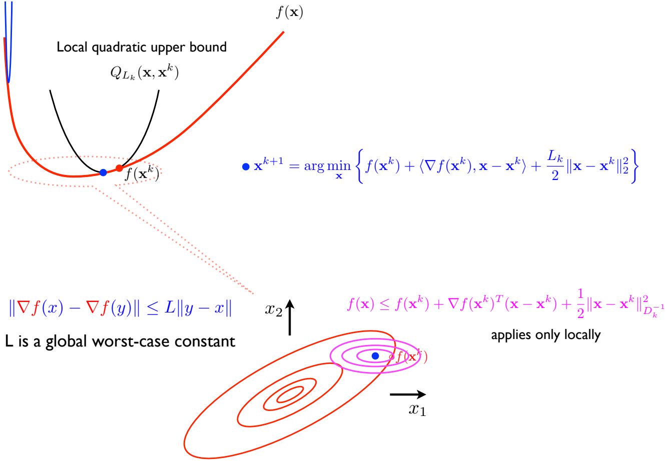
Adaptation through first-order information

- 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 $\mathbf{H}_k = \mathbf{D}_k$ [32]

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

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



AdaGrad - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$ [32]

- 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 $\mathbf{H}_k = \mathbf{D}_k$ [32]

- 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 [15]

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 [16]

Motivation behind AcceleGrad

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

- The answer is yes! AcceleGrad combines an accelerated algorithm with AdaGrad step-size.
- 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
- $\eta_k = \dfrac{2D}{\sqrt{G^2 + \sum_{i=0}^{k} (\alpha_k)^2 \|\nabla f(\mathbf{x}^k)\|^2}}$.

Convergence of AcceleGrad

Theorem (Convergence rate of AcceleGrad)

Let the sequence $\{\mathbf{y}^k\}$ *be generated by AcceleGrad. Under the assumptions*

- f *is convex and* L*-smooth,*
- *Iterates are bounded, such that* $D = \max_{\mathbf{x},\mathbf{y}\in\mathbb{R}^d} \|\mathbf{x}-\mathbf{y}\|$*,*
- *Gradient norms are bounded* $\|\nabla f(\mathbf{x})\| \leq G$*,*

AcceleGrad has the following guarantee:

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

where $\bar{\mathbf{y}}^k = (\sum_{i=0}^{k-1} \alpha_k \mathbf{y}^{k+1})/(\sum_{i=0}^{k-1} \alpha_k)$ *is the average iterate.*

Remarks:

- ◦ Accelegrad is a nearly "universal" algorithm (more on this later!)
- ◦ We still need a bound on G and D to run the algorithm.
- ◦ It cannot handle constraints.

UniXGrad - Accelerated Extra-gradient (!) algorithm for constraints [11]

◦ Universal extra-gradient method offers improvements over AcceleGrad

Extra-Gradient algorithm
1. Choose $\mathbf{x}^0 \in \mathrm{dom}\,(f)$
2. For $k = 0, 1, \ldots,$ iterate
$\begin{cases} \mathbf{x}^{k+1/2} & = \mathbf{x}^k - \alpha \nabla f(\mathbf{x}^k) \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha \nabla f(\mathbf{x}^{k+1/2}) \end{cases}$

UniXGrad
1. Set $\mathbf{x}^0 = \mathbf{z}^0 = \mathbf{x}^0$
2. For $k = 0, 1, \ldots,$ iterate
$\begin{cases} \mathbf{x}^{k+1/2} & = \Pi_{\mathcal{X}} \left(\mathbf{x}^k - \alpha_k \eta_k \nabla f(\tilde{\mathbf{x}}^k)\right) \\ \mathbf{x}^{k+1} & = \Pi_{\mathcal{X}} \left(\mathbf{x}^k - \alpha_k \eta_k \nabla f(\bar{\mathbf{x}}^{k+1/2})\right) \end{cases}$

- Pick $\alpha < 1/L$.

- $\Pi_{\mathcal{X}}(\mathbf{x})$ is Euclidean projection onto $\mathcal{X}$ and $\alpha_k = k$
- $\tilde{\mathbf{x}}^k = \frac{\alpha_k \mathbf{x}^k + \sum_{i=1}^{k-1} \alpha_i \mathbf{x}^{i+1/2}}{\sum_{i=1}^{k} \alpha_i}, \quad \bar{\mathbf{x}}^{k+1/2} = \frac{\sum_{i=1}^{k} \alpha_i \mathbf{x}^{i+1/2}}{\sum_{i=1}^{k} \alpha_i}$
- $\eta_k = \frac{2D}{\sqrt{1 + \sum_{i=1}^{k} (\alpha_k)^2 \|\nabla f(\bar{\mathbf{x}}^{k+1/2}) - \nabla f(\tilde{\mathbf{x}}^k)\|^2}}$

Convergence of UniXGrad

Theorem (Convergence rate of UniXGrad)

Let the sequence $\{\mathbf{x}^{k+1/2}\}$ be generated by UniXGrad. Under the assumptions

- f *is convex and* L*-smooth,*
- *Constraint set* $\mathcal{X}$ *has bounded diameter, i.e.,* $D = \max_{\mathbf{x},\mathbf{y}\in\mathcal{X}} \|\mathbf{x}-\mathbf{y}\|$,

UniXGrad guarantees the following:

$$f(\bar{\mathbf{x}}^{k+1/2}) - \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \leq O\left(\frac{LD^2}{k^2}\right),$$

where $\bar{\mathbf{x}}^{k+1/2} = \frac{\sum_{i=1}^{k} \alpha_i \mathbf{x}^{i+1/2}}{\sum_{i=1}^{k} \alpha_i}$ *is the average iterate.*

Remarks:

- UniXGrad is a truly "universal" algorithm (more on this later!)
- We still need a bound on D to run the algorithm.
- It can handle constraints.
- It removes the $\log$-factor in AcceleGrad.

Adaptive methods and open questions

Question: ◦ Can we improve diameter D dependence on adaptive methods?

Answer: ◦ UnderGrad [3] has $O(\log D)$ dependence instead of $\mathcal{O}(D)$ while retaining the fast rates.

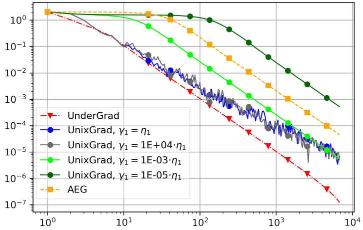


Figure: UniXGrad vs. UnderGrad vs. Accelerated extra-gradient algorithm.

Question: ◦ Can we go beyond $O(1/k^2)$ rate while adapting to problem parameters and oracle noise?

Answer: ◦ Yes, ExtraNewton$^{\text{TM}}$ achieves a rate of $O\left(\frac{1}{k^3}\right)$ using a regularized Newton update.

A quick look at descent methods: beyond first-order minimization

Revisiting majorization-minimization

- Gradient descent, for $\alpha > 0$:

$$\begin{aligned}\mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}\in\mathbb{R}^d} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k\rangle + \tfrac{1}{2\alpha}\|\mathbf{x} - \mathbf{x}^k\|^2 \right\} \\ &= \mathbf{x}^k - \alpha \nabla f(\mathbf{x}^k).\end{aligned}$$

- Newton's method, for $\alpha > 0$:

$$\begin{aligned}\mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}\in\mathbb{R}^d} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k\rangle + \tfrac{1}{2\alpha}\langle \nabla^2 f(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k), \mathbf{x} - \mathbf{x}^k\rangle \right\} \\ &= \mathbf{x}^k - \alpha (\nabla^2 f(\mathbf{x}^k))^{-1} \nabla f(\mathbf{x}^k).\end{aligned}$$

- Regularized Newton's method, for $\alpha, \beta > 0$ [14, 18]:

$$\begin{aligned}\mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}\in\mathbb{R}^d} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k\rangle + \tfrac{1}{2\alpha}\langle \nabla^2 f(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k), \mathbf{x} - \mathbf{x}^k\rangle + \tfrac{1}{2\alpha\beta}\|\mathbf{x} - \mathbf{x}^k\|^2 \right\} \\ &= \mathbf{x}^k - \alpha (\nabla^2 f(\mathbf{x}^k) + \beta\mathbb{I})^{-1} \nabla f(\mathbf{x}^k).\end{aligned}$$

A quick look at descent methods: beyond first-order minimization

Revisiting majorization-minimization

- Gradient descent, for $\alpha > 0$:

$$\begin{aligned}\mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}\in\mathbb{R}^d} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x}-\mathbf{x}^k\rangle + \tfrac{1}{2\alpha}\|\mathbf{x}-\mathbf{x}^k\|^2 \right\} \\ &= \mathbf{x}^k - \alpha\nabla f(\mathbf{x}^k).\end{aligned}$$

- Newton's method, for $\alpha > 0$:

$$\begin{aligned}\mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}\in\mathbb{R}^d} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x}-\mathbf{x}^k\rangle + \tfrac{1}{2\alpha}\langle \nabla^2 f(\mathbf{x}^k)(\mathbf{x}-\mathbf{x}^k), \mathbf{x}-\mathbf{x}^k\rangle \right\} \\ &= \mathbf{x}^k - \alpha(\nabla^2 f(\mathbf{x}^k))^{-1}\nabla f(\mathbf{x}^k).\end{aligned}$$

- Regularized Newton's method, for $\alpha, \beta > 0$ [14, 18]:

$$\begin{aligned}\mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}\in\mathbb{R}^d} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x}-\mathbf{x}^k\rangle + \tfrac{1}{2\alpha}\langle \nabla^2 f(\mathbf{x}^k)(\mathbf{x}-\mathbf{x}^k), \mathbf{x}-\mathbf{x}^k\rangle + \tfrac{1}{2\alpha\beta}\|\mathbf{x}-\mathbf{x}^k\|^2 \right\} \\ &= \mathbf{x}^k - \alpha(\nabla^2 f(\mathbf{x}^k) + \beta\mathbb{I})^{-1}\nabla f(\mathbf{x}^k).\end{aligned}$$

Remarks:

- ◦ Global convergence of the Newton method is difficult.
- ◦ Local convergence of the Newton method using self-concordance is well-studied.
- ◦ Quasi-Newton methods that approximate the Newton method are well-studied [31].
- ◦ See advanced material at the end of the lecture.

ExtraNewton: Adaptive Newton's method with fast rates

Question: ◦ Under what minimal regularity conditions, can we achieve faster rates beyond $O(1/k^2)$?

Answer: ◦ Higher-order smoothness

Second-order smoothness

If the objective f has L-Lipschitz continuous Hessian, then

$$\left| f(\mathbf{x}) - f(\mathbf{y}) - \langle \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle - \frac{1}{2} \langle \nabla^2 f(\mathbf{y})(\mathbf{x} - \mathbf{y}), \mathbf{x} - \mathbf{y} \rangle \right| \leq \frac{L}{6} \|\mathbf{x} - \mathbf{y}\|^3$$

Question: ◦ How can we exploit the higher-order smoothness?

Answer: ◦ Proximal Point method (PPM) + Newton-type updates!

Better approximation, better rates

∘ The extra-gradient method approximates PPM through the “extrapolation” sequence $\mathbf{x}^{k+1/2}$ [21]

Higher-order information for better approximation

- Extra-gradient approximates the **“next”** iterate, $\mathbf{x}^{k+1}$, using first-order information.
- Can we achieve a better estimate $\mathbf{x}^{k+1/2}$ using second-order information? **YES!**

ExtraNewton [2]

1. Set $\mathbf{x}^0 = \mathbf{z}^0 = \mathbf{x}^0$. Define $\alpha_k = k^2$ and $A_k = \sum_{i=1}^{k} \alpha_k$
2. For $k = 0, 1, \ldots$, iterate

$$\begin{cases} \mathbf{x}^{k+1/2} & = \mathbf{x}^k - \alpha_k \eta_k \left(\eta_k \frac{\alpha_k^2}{A_k} \nabla^2 f(\tilde{\mathbf{x}}^k) + \mathbb{I} \right)^{-1} \nabla f(\tilde{\mathbf{x}}^k) \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \eta_k \nabla f(\bar{\mathbf{x}}^{k+1/2}) \end{cases}$$

- $\tilde{\mathbf{x}}^k = \dfrac{\alpha_k \mathbf{x}^k + \sum_{i=1}^{k-1} \alpha_i \mathbf{x}^{i+1/2}}{\sum_{i=1}^{k} \alpha_i}, \quad \bar{\mathbf{x}}^{k+1/2} = \dfrac{\sum_{i=1}^{k} \alpha_i \mathbf{x}^{i+1/2}}{\sum_{i=1}^{k} \alpha_i},$
- $\eta_k = \dfrac{\gamma}{\sqrt{1 + \sum_{i=1}^{k-1} (\alpha_k)^2 \|\nabla f(\bar{\mathbf{x}}^{k+1/2}) - \nabla f(\tilde{\mathbf{x}}^k) - \nabla^2 f(\tilde{\mathbf{x}}^k)(\bar{\mathbf{x}}^{k+1/2} - \tilde{\mathbf{x}}^k)\|^2}}.$

Convergence of ExtraNewton

Theorem ([2])

Let the sequence $\mathbf{x}^{k+1/2}$ be generated by ExtraNewton. Under the assumptions

- f *has* L*-Lipschitz Hessian (not Lipschitz* **smooth***),*
- $D = \max_{\mathbf{x},\mathbf{y}\in\mathcal{X}} \|\mathbf{x}-\mathbf{y}\|$

ExtraNewton guarantees that

$$f(\bar{\mathbf{x}}^{k+1/2}) - \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \leq O\left(\frac{L\left(\frac{D^4}{\gamma} + D\gamma^2\right)}{k^3}\right),$$

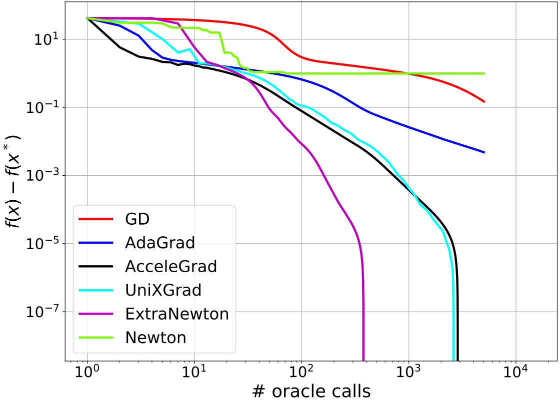
where $\bar{\mathbf{x}}^{k+1/2} = \frac{\sum_{i=1}^k \alpha_i \mathbf{x}^{i+1/2}}{\sum_{i=1}^k \alpha_i}$ *is the average sequence.*

Remarks:

- ◦ The first globally convergent Newton method without a line-search procedure.
- ◦ The algorithm does not need to know the diameter D.
- ◦ ExtraNewton is also noise-adaptive; continuously adapts to noise in oracles.

Logistic regression: ExtraNewton vs. adaptive first-order methods

- Logistic regression with regularization using a1a dataset.
- Comparison against first-order adaptive methods.



Lower bounds for higher-order smoothness?

- Higher-order methods and the limits of their performance has received great attention lately.
- Beyond Lipschitz smoothness, we can achieve improving sublinear rates.

Theorem ([27])

Consider that f is p-th order smooth (equivalently has Lipschitz continuous $p+1$-th order derivative). Let $\mathbf{x}^k$ be generated by some p-th order iterative tensor method. Then, it holds that

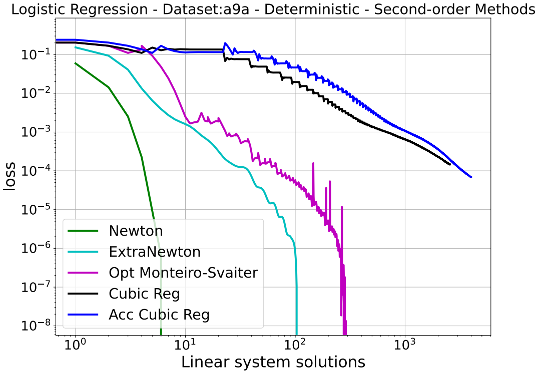
$$\min_{0\leq i\leq k} f(\mathbf{x}^i) - \min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) = \Omega\left(\frac{1}{k^{\frac{3p+1}{2}}}\right).$$

Remarks:

- AGD matches the lower bound for 1-st order smooth function.
- The lower bound for second-order methods evaluates to $O\left(\frac{1}{k^{7/2}}\right)$.
- Monteiro-Svaiter's accelerated Newton method [22] and a recent work [5] archive this rate.
- In practice, all of them seem slower than ExtraNewton.

Logistic regression: ExtraNewton vs. second-order methods

- Logistic regression with regularization using a9a dataset.
- Comparison against second-order methods with matching and optimal rates.



- Legend:
  - Optimal Monteiro-Svaiter [5],
  - Cubic regularization of Newton's method [25],
  - Accelerated cubic regularization of Newton's methods [24].

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
	Prodigy [20]	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	AcceleGrad	Sublinear $(1/k^2)$	One gradient
	UniXGrad	Sublinear $(1/k^2)$	Two gradients
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
	Reg. Newton method	Sublinear $(1/k^2)$	One gradient, one linear system
	ExtraNewton method	Sublinear $(1/k^3)$	Two gradients, 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
	Sophia [17]	Linear (e^{-k})	SVD dec., one linear system, one gradient

Gradient descent:

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

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

AdaGrad:

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

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

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
	Prodigy [20]	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	AcceleGrad	Sublinear $(1/k^2)$	One gradient
	UniXGrad	Sublinear $(1/k^2)$	Two gradients
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
	Reg. Newton method	Sublinear $(1/k^2)$	One gradient, one linear system
	ExtraNewton method	Sublinear $(1/k^3)$	Two gradients, 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
	Sophia [17]	Linear (e^{-k})	SVD dec., one linear system, one gradient

UniXGrad:

$$\mathbf{x}^{k+1/2} = \mathbf{x}^k - \alpha_k \eta_k \nabla f(\tilde{\mathbf{x}}^k)$$
$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \eta_k \nabla f(\bar{\mathbf{x}}^{k+1/2}).$$

for some proper choice of $\alpha_k = k$ and η_k.

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

$$\eta_k = 2D\left(G^2 + \sum_{i=0}^{k} (\alpha_k)^2 \|\nabla f(\mathbf{x}^k)\|^2\right)^{-1/2}.$$

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
	Prodigy	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	AcceleGrad	Sublinear $(1/k^2)$	One gradient
	UniXGrad	Sublinear $(1/k^2)$	Two gradients
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
	Reg. Newton method	Sublinear $(1/k^2)$	One gradient, one linear system
	ExtraNewton method	Sublinear $(1/k^3)$	Two gradients, 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
	Sophia	Linear (e^{-k})	SVD dec., one linear system, one gradient

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

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

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
	Prodigy	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	AcceleGrad	Sublinear $(1/k^2)$	One gradient
	UniXGrad	Sublinear $(1/k^2)$	Two gradients
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
	Reg. Newton method	Sublinear $(1/k^2)$	One gradient, one linear system
	ExtraNewton method	Sublinear $(1/k^3)$	Two gradients, 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
	Sophia	Linear (e^{-k})	SVD dec., one linear system, one gradient

Prodigy [20] is gradient descent with step sizes defined as

$$\gamma_t = \frac{d_t^2}{\sqrt{d_t^2 G^2 + \sum_{i=0}^{k} d_i^2 \| \nabla f(x_i) \|^2}} \quad \text{with} \quad d_{i+1} = \max\left\{ d_i, \frac{\sum_{i=0}^{k} \gamma_i \langle g_i, x_0 - x_i \rangle}{\| x_{i+1} - x_0 \|} \right\}$$

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
	Prodigy	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	AcceleGrad	Sublinear $(1/k^2)$	One gradient
	UniXGrad	Sublinear $(1/k^2)$	Two gradients
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
	Reg. Newton method	Sublinear $(1/k^2)$	One gradient, one linear system
	ExtraNewton method	Sublinear $(1/k^3)$	Two gradients, 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
	Sophia	Linear (e^{-k})	SVD dec., one linear system, one gradient

- Sophia stands for Second-order Clipped Stochastic Optimization [17].
- They introduce a novel Hessian estimator to stabilize the Newton's method in nonconvex landscapes.

Overview of adaptive methods

A non-exhaustive comparison:

Assumptions on f	**Algorithm**	**Convergence rate**	**Setting**
L-smooth	AdaGrad	Sublinear $(1/k)$	Unknown L
	Prodigy	Sublinear $(1/k)$	Unknown $\| x - x_0 \|$
	AcceleGrad	Sublinear $(1/k^2)$	Unknown L
	UniXGrad	Sublinear $(1/k^2)$	Unknown L
	ExtraNewton method	Sublinear $(1/k^3)$	Unknown L, $\| x - x_0 \|$

◦ Notice that L and $\| x - x_0 \|$ are rarely known in real world problems.

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 [4]

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.

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**.

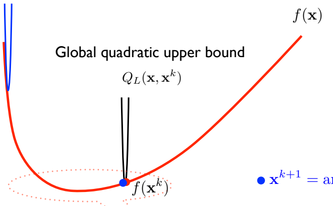
Line-search

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 $c \in (0, 1]$.

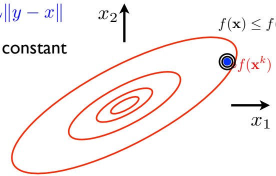
⋆How can we better adapt to the local geometry?



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

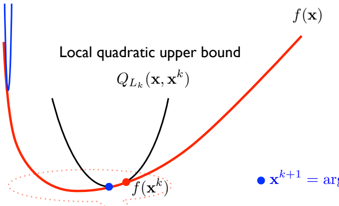
$$\|\nabla f(x) - \nabla f(y)\| \leq L\|y - x\|$$

L is a global worst-case constant



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

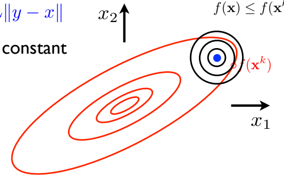
⋆How can we better adapt to the local geometry?



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

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

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?

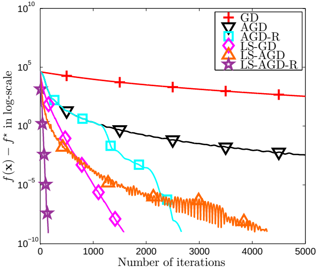
- AGD-μL requires knowledge of μ and AGD-L does not have optimal convergence for strongly convex f.
- AGD is non-monotonic (i.e., $f(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k)$ is not always satisfied).
- AGD has a periodic behavior, where the momentum depends on the local condition number $\kappa = L/\mu$.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

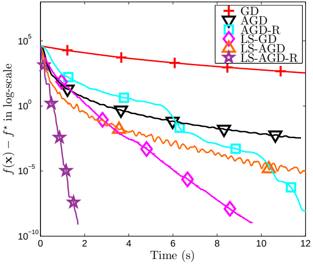
Restart strategies

1. **O'Donoghue - Candes's strategy [29]:** 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 [10]:** Do not require $t_k = 1$ and do not necessary require function evaluations.
3. **Fercoq-Qu's strategy [9]:** Unconditional periodic restart for strongly convex functions. Do not require the strong convexity parameter.

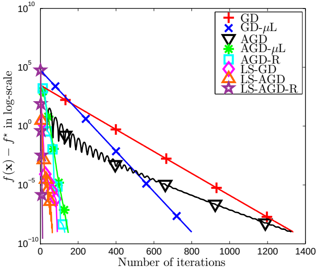
⋆Example: Ridge regression

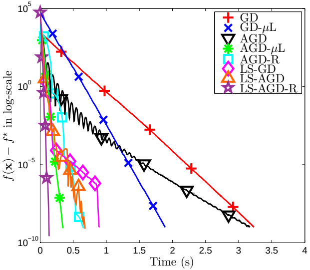
Case 1: $n = 500, p = 2000, \rho = 0$





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





*AcceleGrad - Adaptive gradient + Accelerated gradient [16]

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 $\to$ 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

RMSProp + 2nd order moment estimation = 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 [30].

∘ An ADAM alternative is proposed that is proved to be convergent [30].

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 $\left\{\begin{array}{ll} \mathbf{g}_k & = G(\mathbf{x}^k, \theta) \\ \mathbf{m}_k & = \beta_{1,k}\mathbf{m}_{k-1} + (1-\beta_{1,k})\mathbf{g}_k \leftarrow \text{1st order estimate} \\ \mathbf{v}_k & = \beta_2\mathbf{v}_{k-1} + (1-\beta_2)\mathbf{g}_k^2 \leftarrow \text{2nd order estimate} \\ \hat{\mathbf{v}}_k & = \max\{\hat{\mathbf{v}}_{k-1}, \mathbf{v}_k\} \text{ and } \hat{\mathbf{V}}_k = \mathrm{diag}(\hat{\mathbf{v}}_k) \\ \mathbf{H}_k & = \sqrt{\hat{\mathbf{v}}_k} \\ \mathbf{x}^{k+1} & = \Pi_{\mathcal{X}}^{\sqrt{\hat{\mathbf{V}}_k}}(\mathbf{x}^k - \alpha_k\hat{\mathbf{m}}_k/\mathbf{H}_k) \end{array}\right.$
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)

The key ingredient of acceleration: (weighted) averaging

- One common theme we see in acceleration schemes is iterate averaging.
- It is important to compute averages with larger weights on recent iterates.
- Through UniXGrad/Extra-gradient framework, we could summarize the effect of averaging.

Convergence rate vs. averaging parameter

Let $\{\mathbf{x}^{k+1/2}\}$ be a sequence generated by UniXGrad algorithm, and define $0 < \alpha_k < O(k)$ to be a non-decreasing sequence of weights. It is ensured that,

$$f(\bar{\mathbf{x}}^{k+1/2}) - \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \leq O\left(\frac{1}{\sum_{i=1}^{k} \alpha_k}\right)$$

The key ingredient of acceleration: (weighted) averaging

- One common theme we see in acceleration schemes is iterate averaging.
- It is important to compute averages with larger weights on recent iterates.
- Through UniXGrad/Extra-gradient framework, we could summarize the effect of averaging.

Convergence rate vs. averaging parameter

Let $\{\mathbf{x}^{k+1/2}\}$ be a sequence generated by UniXGrad algorithm, and define $0 < \alpha_k < O(k)$ to be a non-decreasing sequence of weights. It is ensured that,

$$f(\bar{\mathbf{x}}^{k+1/2}) - \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \leq O\left(\frac{1}{\sum_{i=1}^{k} \alpha_k}\right)$$

Remarks:

- Uniform averaging: $\alpha_k = 1 \implies O\left(\frac{1}{k}\right)$ convergence rate
- Weighted averaging: $\alpha_k = O(k) \implies O\left(\frac{1}{k^2}\right)$ convergence rate
- In general: $\alpha_k = O(k^p)$ for $p \in [0, 1] \implies O\left(\frac{1}{k^{p+1}}\right)$

*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 [28]

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

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

*Locally quadratic convergence of the Newton method–II

Newton's method local quadratic convergence - Proof [28].

- Recall

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

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

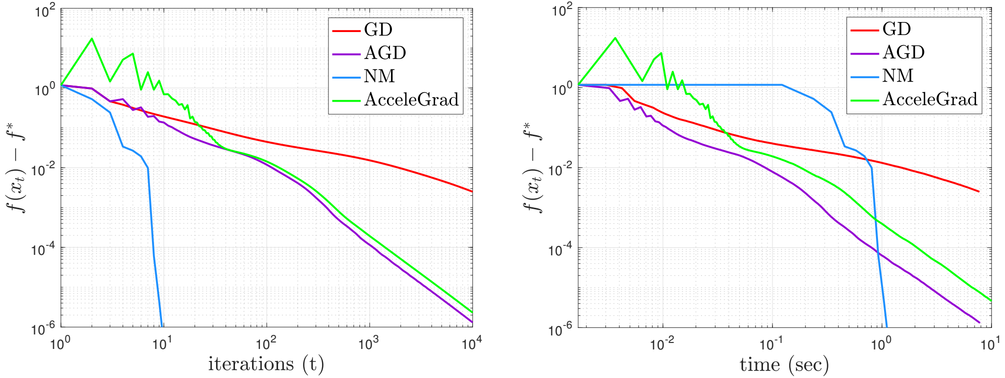
- Since $\nabla^2 f(\mathbf{x}^\star)$ is nonsingular, there must exist a radius r such that $\|(\nabla^2 f(\mathbf{x}^k))^{-1}\| \leq 2\|(\nabla^2 f(\mathbf{x}^\star))^{-1}\|$ for all $\mathbf{x}^k$ with $\|\mathbf{x}^k - \mathbf{x}^*\| \leq r$.
- Substituting, we obtain

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

  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 $\mathbf{x}^\star$.

□

*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 [6].
- For more details on quasi-Newton methods, see Nocedal&Wright's book [28].

*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 [28] (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 [28] (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

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

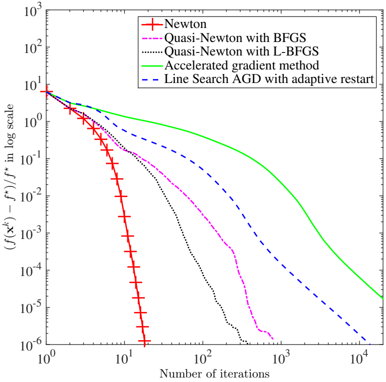
⋆L-BFGS: A quasi-Newton method

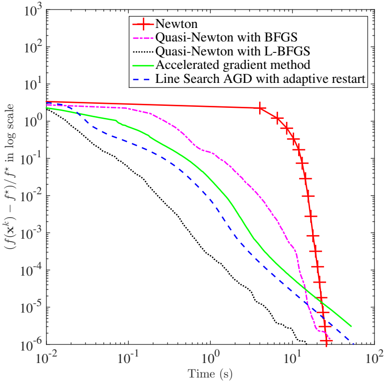
L-BFGS
1. Choose starting point $\mathbf{x}^0$ and $m > 0$. **2.** For $k = 0, 1, \ldots$ **2.a** Choose $\mathbf{H}_k^0$. **2.b** Compute $\mathbf{p}^k = -\mathbf{H}_k \nabla f(\mathbf{x}^k)$ using the previous algorithm. **2.c** Set $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}^k$, where α_k 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 $\mathbf{x}^\star$, $f^\star$ by running Newton's method for 200 iterations.

*Tensor methods I

∘ Let us investigate a generic method for handling p-th order smooth problems using p-th order derivatives.

Taylor polynomial

Let us focus on the Taylor polynomial expansion for a function $f(\mathbf{x})$ of order p at $\mathbf{x}$:

$$T_p(\mathbf{x};\mathbf{y}) = f(\mathbf{x}) + \sum_{i=1}^{p} \frac{1}{i!} D^i f(\mathbf{x})[\mathbf{y}-\mathbf{x}]^i,$$

- $D^i f(\mathbf{x})[h]^i$ is the directional derivative along h such that

$$D^1 f(\mathbf{x})[h] = \langle \nabla f(\mathbf{x}), h\rangle, \qquad \text{and} \qquad D^2 f(\mathbf{x})[h]^2 = \langle \nabla^2 f(\mathbf{x})h, h\rangle,$$

- p-th order smoothness:

$$|f(\mathbf{y}) - T_p(\mathbf{x},\mathbf{y})| \leq \frac{L_p}{(p+1)!}\|\mathbf{x}-\mathbf{y}\|^{p+1},$$

- Regularized Taylor polynomial of order p at $\mathbf{x}$:

$$\hat{T}_{p,H}(\mathbf{x};\mathbf{y}) = f(\mathbf{x}) + \sum_{i=1}^{p} \frac{1}{i!} D^i f(\mathbf{x})[\mathbf{y}-\mathbf{x}]^i + \frac{pH}{(p+1)!}\|\mathbf{x}-\mathbf{x}^k\|^{p+1}.$$

*Tensor methods I

∘ Let us investigate a generic method for handling p-th order smooth problems using p-th order derivatives.

Taylor polynomial

Let us focus on the Taylor polynomial expansion for a function $f(\mathbf{x})$ of order p at $\mathbf{x}$:

$$T_p(\mathbf{x};\mathbf{y}) = f(\mathbf{x}) + \sum_{i=1}^{p} \frac{1}{i!} D^i f(\mathbf{x})[\mathbf{y}-\mathbf{x}]^i,$$

- $D^i f(\mathbf{x})[h]^i$ is the directional derivative along h such that

$$D^1 f(\mathbf{x})[h] = \langle \nabla f(\mathbf{x}), h\rangle, \qquad \text{and} \qquad D^2 f(\mathbf{x})[h]^2 = \langle \nabla^2 f(\mathbf{x})h, h\rangle,$$

- p-th order smoothness:

$$|f(\mathbf{y}) - T_p(\mathbf{x},\mathbf{y})| \le \frac{L_p}{(p+1)!}\|\mathbf{x}-\mathbf{y}\|^{p+1},$$

- Regularized Taylor polynomial of order p at $\mathbf{x}$:

$$\hat{T}_{p,H}(\mathbf{x};\mathbf{y}) = f(\mathbf{x}) + \sum_{i=1}^{p} \frac{1}{i!} D^i f(\mathbf{x})[\mathbf{y}-\mathbf{x}]^i + \frac{pH}{(p+1)!}\|\mathbf{x}-\mathbf{x}^k\|^{p+1}.$$

Remark: ∘ If $H \ge L_p$, then, $f(\mathbf{y}) \le \hat{T}_p(\mathbf{x};\mathbf{y})$ and $\hat{T}_p(\mathbf{x};\mathbf{y})$ is convex. We will assume this condition!

*Tensor methods II

◦ Recall regularized Taylor polynomial of order p at $\mathbf{x}^k$:

$$\hat{T}_{p,H}(\mathbf{x};\mathbf{y}) = f(\mathbf{x}) + \sum_{i=1}^{p} \frac{1}{i!} D^i f(\mathbf{x})[\mathbf{y}-\mathbf{x}]^i + \frac{pH}{(p+1)!}\|\mathbf{x}-\mathbf{x}^k\|^{p+1}.$$

Approach: ◦ Use $\hat{T}_{p,H}(\mathbf{x}^k;\mathbf{x})$ as the new majorizer, and minimize to obtain $\mathbf{x}^{k+1}$

Tensor method [27]
1. Choose $\mathbf{x}^0 = \mathbf{y}^0 \in \mathrm{dom}\,(f)$
2. For $k = 0, 1, \ldots,$ iterate
$\left\{ \mathbf{x}^{k+1} = \arg\min_{\mathbf{x}\in\mathbb{R}^d} \hat{T}_{p,H}(\mathbf{x}^k;\mathbf{x}) \right.$

Theorem (Convergence of p-th order tensor method [27])

Consider f to be p-th order smooth and let $\{\mathbf{x}^k\}$ be generated by the Tensor method. Then, it holds that

$$f(\mathbf{x}^k) - \min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) \le O\left(\frac{1}{t^p}\right).$$

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`.
(Cited on page 63.)

[2] Kimon Antonakopoulos, Ali Kavis, and Volkan Cevher.
A first approach to universal second-order acceleration for convex minimization.
Advances In Neural Information Processing Systems 35 (Nips 2022), (CONF), 2022.
(Cited on pages 42 and 43.)

[3] Kimon Antonakopoulos, Dong Quan Vu, Volkan Cevher, Kfir Levy, and Panayotis Mertikopoulos.
UnderGrad: A universal black-box optimization method with almost dimension-free convergence rate guarantees.
In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 772–795. PMLR, 17–23 Jul 2022.
(Cited on page 38.)

References II

[4] Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford.
Lower bounds for finding stationary points i.
arXiv preprint arXiv:1710.11606, 2017.
(Cited on page 54.)

[5] Yair Carmon, Danielle Hausler, Arun Jambulapati, Yujia Jin, and Aaron Sidford.
Optimal and adaptive monteiro-svaiter acceleration.
CoRR, abs/2205.15371, 2022.
(Cited on pages 45 and 46.)

[6] JE Dennis and Jorge J Moré.
A characterization of superlinear convergence and its application to quasi-newton methods.
Math. Comp., 28(126):549–560, 1974.
(Cited on page 83.)

[7] Jelena Diakonikolas and Lorenzo Orecchia.
Accelerated extra-gradient descent: A novel accelerated first-order method.
2018.
(Cited on pages 15 and 16.)

References III

[8] John Duchi, Elad Hazan, and Yoram Singer.
Adaptive subgradient methods for online learning and stochastic optimization.
J. Mach. Learn. Res., 12(null):2121–2159, July 2011.
(Cited on pages 26 and 27.)

[9] Olivier Fercoq and Zheng Qu.
Restarting accelerated gradient methods with a rough strong convexity estimate.
arXiv preprint arXiv:1609.07358, 2016.
(Cited on page 61.)

[10] P. Giselsson and S. Boyd.
Monotonicity and Restart in Fast Gradient Methods.
In *IEEE Conf. Decision and Control*, Los Angeles, USA, December 2014. CDC.
(Cited on page 61.)

[11] Ali Kavis, Kfir Y Levy, Francis Bach, and Volkan Cevher.
Unixgrad: A universal, adaptive algorithm with optimal guarantees for constrained optimization.
Advances In Neural Information Processing Systems 32 (Nips 2019), 32(CONF), 2019.
(Cited on page 36.)

References IV

[12] Diederik P Kingma and Jimmy Ba.
Adam: A method for stochastic optimization.
arXiv preprint arXiv:1412.6980, 2014.
(Cited on pages 26 and 27.)

[13] Galina M Korpelevich.
The extragradient method for finding saddle points and other problems.
Matecon, 12:747–756, 1976.
(Cited on pages 15 and 16.)

[14] Kenneth Levenberg.
A method for the solution of certain non – linear problems in least squares.
Quarterly of Applied Mathematics, 2:164–168, 1944.
(Cited on pages 39 and 40.)

[15] Kfir Levy.
Online to offline conversions, universality and adaptive minibatch sizes.
In *Advances in Neural Information Processing Systems*, pages 1613–1622, 2017.
(Cited on pages 28, 29, and 33.)

References V

[16] 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.
(Cited on pages 34 and 63.)

[17] Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma.
Sophia: A scalable stochastic second-order optimizer for language model pre-training.
arXiv preprint arXiv:2305.14342, 2023.
(Cited on pages 48, 49, and 52.)

[18] Donald W. Marquardt.
An algorithm for least-squares estimation of nonlinear parameters.
Journal of the Society for Industrial and Applied Mathematics, 11(2):431–441, 1963.
(Cited on pages 39 and 40.)

[19] H. Brendan McMahan and Matthew Streeter.
Adaptive Bound Optimization for Online Convex Optimization.
In *Proceedings of the 23rd Annual Conference on Learning Theory (COLT)*, 2010.
(Cited on pages 26, 27, 28, and 29.)

References VI

[20] Konstantin Mishchenko and Aaron Defazio.
Prodigy: An expeditiously adaptive parameter-free learner.
arXiv preprint arXiv:2306.06101, 2023.
(Cited on pages 48, 49, and 51.)

[21] Aryan Mokhtari, Asuman Ozdaglar, and Sarath Pattathil.
A unified analysis of extra-gradient and optimistic gradient methods for saddle point problems: Proximal point approach.
In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1497–1507. PMLR, 26–28 Aug 2020.
(Cited on page 42.)

[22] Renato D. C. Monteiro and B. F. Svaiter.
An accelerated hybrid proximal extragradient method for convex optimization and its implications to second-order methods.
SIAM Journal on Optimization, 23(2):1092–1125, 2013.
(Cited on page 45.)

References VII

[23] Y. Nesterov.
Introductory lectures on convex optimization: A basic course, volume 87.
Springer, 2004.
(Cited on pages 5, 11, and 12.)

[24] Y. Nesterov.
Accelerating the cubic regularization of Newton's method on convex problems.
Math. Program., 112:159–181, 2008.
(Cited on page 46.)

[25] Y. Nesterov and B.T. Polyak.
Cubic regularization of newton method and its global performance.
Math. Program., 108(1):177–205, 2006.
(Cited on page 46.)

[26] Yu. E. Nesterov.
A method of solving a convex programming problem with convergence rate $O(1/k^2)$.
Soviet Math. Dokl., 27(2):372–376, 1983.
(Cited on page 14.)

References VIII

[27] Yurii Nesterov.
Implementable tensor methods in unconstrained convex optimization.
Math. Program., 186(1–2):157–183, mar 2021.
(Cited on pages 45 and 93.)

[28] J. Nocedal and S. Wright.
Numerical Optimization.
Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.
(Cited on pages 80, 81, 83, 85, and 86.)

[29] Brendan O'donoghue and Emmanuel Candes.
Adaptive restart for accelerated gradient schemes.
Foundations of computational mathematics, 15(3):715–732, 2015.
(Cited on page 61.)

[30] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar.
On the convergence of adam and beyond.
arXiv preprint arXiv:1904.09237, 2019.
(Cited on page 71.)

References IX

[31] Anton Rodomanov.
Quasi-Newton methods with provable efficiency guarantees.
PhD thesis, UCL - Université Catholique de Louvain, 2022.
(Cited on pages 39 and 40.)

[32] Matthew J. Streeter and H. Brendan McMahan.
Less Regret via Online Conditioning.
CoRR, abs/1002.4862, 2010.
arXiv: 1002.4862.
(Cited on pages 30, 31, and 32.)

[33] 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.
(Cited on pages 26 and 27.)