Mathematics of Data: From Theory to Computation

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 5: Unconstrained, smooth minimization III

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

EE-556 (Fall 2019)

License Information for Mathematics of Data Slides

Outline

- This lecture
  1. Adaptive gradient methods
  2. Newton's method
  3. Accelerated adaptive gradient methods
- Next lecture
  1. Stochastic gradient methods

Recommended reading

- Chapters 2, 3, 5, 6 in Nocedal, Jorge, and Wright, Stephen J., *Numerical Optimization*, Springer, 2006.
- Chapter 9 in Boyd, Stephen, and Vandenberghe, Lieven, *Convex optimization*, Cambridge university press, 2009.
- Chapter 1 in Bertsekas, Dimitris, *Nonlinear Programming*, Athena Scientific, 1999.
- Chapters 1, 2 and 4 in Nesterov, Yurii, *Introductory Lectures on Convex Optimization: A Basic Course*, Vol. 87, Springer, 2004.

Motivation

Motivation

This lecture covers some more advanced numerical methods for *unconstrained* and *smooth* convex minimization.

Recall: convex, unconstrained, smooth minimization

Problem (**Mathematical formulation**)

$$\boxed{F^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \{F(\mathbf{x}) := f(\mathbf{x})\}} \tag{1}$$

where f is proper, closed, convex and twice differentiable.
Note that (1) is unconstrained.

How de we design efficient optimization algorithms with accuracy-computation tradeoffs for this class of functions?

Recall: Gradient descent methods (convex)

Gradient descent (GD) algorithm

The gradient method we discussed before indeed use the local steepest direction:

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

so that

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

Key question: How do we choose α_k so that we are guaranteed to successfully descend? (ideally as fast as possible)

Answer: By exploiting the structures within the convex function

When $f \in \mathcal{F}_L^{2,1}$, we can use $\alpha_k = 1/L$ so that $\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{1}{L}\nabla f(\mathbf{x}^k)$ is contractive.

- So far, we need to know L to achieve these rates.
- **Another key question:** What if we cannot compute L? Linesearch?
- **One more key question:** Is there any way of automatically exploiting local geometry?

Gradient descent vs. Accelerated gradient descent

Assumptions, step sizes and convergence rates

Gradient descent:

$$f \in \mathcal{F}_L^{2,1}, \quad \alpha = \frac{1}{L} \qquad\qquad f(\mathbf{x}^k) - f(\mathbf{x}^\star) \leq \frac{2L}{k+4}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2.$$

Accelerated Gradient Descent:

$$f \in \mathcal{F}_L^{1,1}, \quad \alpha = \frac{1}{L} \qquad\qquad f(\mathbf{x}^k) - f^\star \leq \frac{4L}{(k+2)^2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2, \ \forall k \geq 0.$$

- 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.

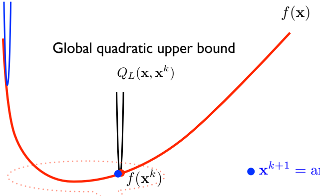
They do so by making use of the information from gradients and their norms.

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?

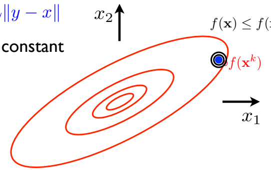


- $\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L}{2} \|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$

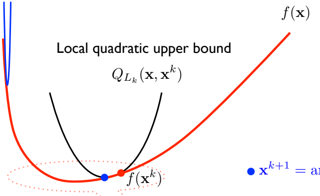
$\|\nabla f(x) - \nabla f(y)\| \leq L\|y - x\|$

L is a global worst-case constant

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



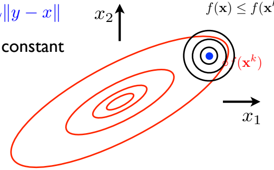
How can we better adapt to the local geometry?



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

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

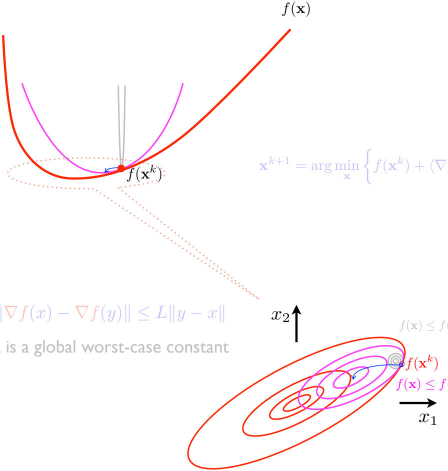
L is a global worst-case constant



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

applies only locally

How can we better adapt to the local geometry?



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

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

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.

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 $g_1, g_2, \cdots g_t$.

- Generally, $\mathbf{H}_k = h_t(g_1, g_2, \cdots, g_t)$ for some mapping h_t
- Some well-known examples:

AdaGrad [3]

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

RmsProp [8]

$$H_k = \sqrt{\beta H_{k-1} + (1-\beta)\mathrm{diag}(\nabla f(x^k))^2}$$

ADAM [4]

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

AdaGrad - Adaptive gradient method with $H_k = \lambda_k I$

- If $H_k = \lambda_k I$, it becomes gradient descent method with adaptive stepsize $\frac{\alpha_k}{\lambda_k}$.

How stepsize adapts?

If gradient $\|\nabla f(x^k)\|$ is large/small $\rightarrow$ AdaGrad adjusts stepsize α_k/λ_k smaller/larger

Adaptive gradient descent(AdaGrad with $H_k = \lambda_k I$) [5]

1. Set $Q_0 = 0$.
2. For $k = 0, 1, \ldots, T$, iterate

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

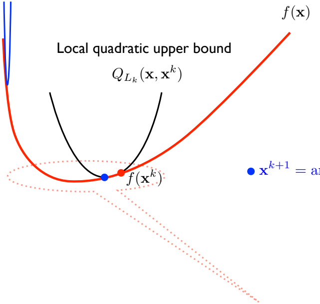
Adaptation through first-order information

- 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 $H_k = D_k$

Adaptation strategy of positive diagonal H_k

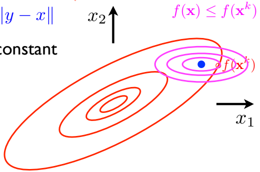
Adaptive stepsize + coordinate-wise extension = adaptive stepsize for each coordinate



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

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

L is a global worst-case constant



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

applies only locally

AdaGrad - Adaptive gradient method with $H_k = D_k$

- Suppose H_k is

$$H_k = \begin{bmatrix} \lambda_{k,1} & & 0 \\ & \ddots & \\ 0 & & \lambda_{k,d} \end{bmatrix},$$

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

Adaptive gradient descent(AdaGrad with $H_k = D_k$)
1. Set $Q_0 = 0$.
2. For $k = 0, 1, \ldots, T$, iterate $\begin{cases} Q^k &= Q^{k-1} + \mathrm{diag}(\nabla f(x^k))^2 \\ H_k &= \sqrt{Q^k} \\ x^{k+1} &= x^k - \alpha_k H_k^{-1} \nabla f(x^k) \end{cases}$

Adaptation across each coordinate

- When $H_k = D_k$, 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 $D = \max_k \|x^k - x^*\|_2$ and $\alpha_k = \frac{D}{\sqrt{2}}$. Define $\bar{\mathbf{x}}^k = (\sum_{i=1}^k \mathbf{x}^i)/t$. Then,

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

A more familiar convergence result [5]

Assume $f \in \mathcal{F}_L^{1,1}$, $D = \max_t \|x^k - x^*\|_2$ and $\alpha_k = \frac{D}{\sqrt{2}}$. Define $\bar{\mathbf{x}}^k = (\sum_{i=1}^k \mathbf{x}^i)/t$. Then,

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

RMSProp - Adaptive gradient method with $H_k = D_k$

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 $H_k = D_k$	RMSProp
1. Set $Q_0 = 0$. **2.** For $k = 0, 1, \ldots, T$, iterate $\begin{cases} Q^k &= Q^{k-1} + \mathrm{diag}(\nabla f(x^k))^2 \\ H_k &= \sqrt{Q^k} \\ x^{k+1} &= x^k - \alpha_k H_k^{-1} \nabla f(x^k) \end{cases}$	**1.** Set $Q_0 = 0$. **2.** For $k = 0, 1, \ldots, T$, iterate $\begin{cases} Q^k &= \beta Q^{k-1} + (1-\beta)\mathrm{diag}(\nabla f(x^k))^2 \\ H_k &= \sqrt{Q^k} \\ x^{k+1} &= x^k - \alpha_k H_k^{-1} \nabla f(x^k) \end{cases}$

- 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
Input. Step size α, exponential decay rates $\beta_1, \beta_2 \in [0, 1)$
1. Set $m_0, V_0 = 0$ **2.** For $k = 0, 1, \ldots, T$, iterate $\begin{cases} g_k &= \nabla f(x^{k-1}) \\ m_k &= \beta_1 m_{k-1} + (1-\beta_1) g_k \leftarrow \text{1st order estimate} \\ v_k &= \beta_2 v_{k-1} + (1-\beta_2) g_k^2 \leftarrow \text{2nd order estimate} \\ \hat{m}_k &= m_k / (1-\beta_1^k) \leftarrow \text{Bias correction} \\ \hat{v}_k &= v_k / (1-\beta_2^k) \leftarrow \text{Bias correction} \\ H_k &= \sqrt{\hat{v}_k} + \epsilon \\ x^{k+1} &= x^k - \alpha \hat{m}_k / H_k \end{cases}$ **3.** Return x^T

(Every vector operation is element-wise operation)

AcceleGrad - Adaptive gradient + Accelerated gradient [6]

Motivation behind AcceleGrad

Is it possible to achieve acceleration for $f \in F_L^{2,1}$, without knowing the Lipschitz constant?

AcceleGrad (Accelerated Adaptive Gradient Method)
Input : Number of iterations T, $x_0 \in \mathcal{K}$, diameter D, weights $\{\alpha_k\}_{k\in[T]}$, learning rate $\{\eta_k\}_{k\in[T]}$
1. Set $y_0 = z_0 = x_0$ **2.** For $k = 0, 1, \ldots, T$, iterate $\begin{cases} \tau_k & := 1/\alpha_k \\ x^{k+1} & = \tau_k z^k + (1-\tau_k)y^k, \text{define } g_k := \nabla f(x^{k+1}) \\ z^{k+1} & = \Pi_{\mathcal{K}}(z^k - \alpha_k \eta_k g_k) \\ y^{k+1} & = x^{k+1} - \eta_k g_k \end{cases}$
Output : $\overline{y}^T \propto \sum_{k=0}^{T-1} \alpha_k y^{k+1}$

- This is essentially the **MD + GD** scheme, with an adaptive step size!

AcceleGrad - Properties and convergence

Learning rate and weight computation

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

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

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

Convergence rate of AcceleGrad

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

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

If f is only convex and G-Lipschitz, then

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

Example: Logistic regression

Problem (Logistic regression)

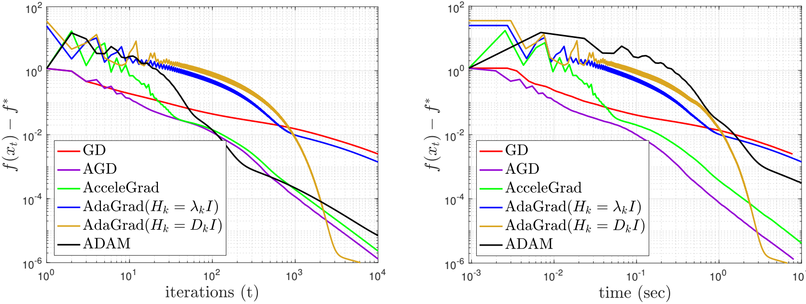
Given $\mathbf{A} \in \{0,1\}^{n \times p}$ *and* $\mathbf{b} \in \{-1,+1\}^n$, *solve:*

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

Real data

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

Example: Logistic regression with adaptive methods



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 [7]

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 [7].

- Recall

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

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

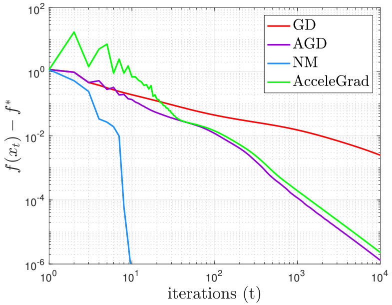
- Since $\nabla^2 f(\mathbf{x}^\star)$ is nonsingular, there must exist a radius r such that $\|(\nabla^2 f(\mathbf{x}^k))^{-1}\| \leq 2\|(\nabla^2 f(\mathbf{x}^\star))^{-1}\|$ for all $\mathbf{x}^k$ with $\|\mathbf{x}^k - \mathbf{x}^*\| \leq r$.
- Substituting, we obtain

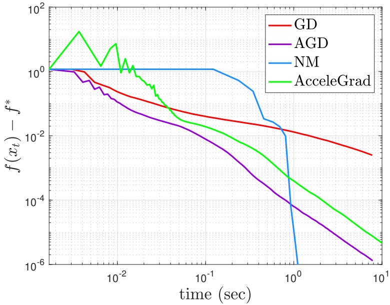
$$\|\mathbf{x}^{k+1} - \mathbf{x}^\star\| \leq M\|(\nabla^2 f(\mathbf{x}^\star))^{-1}\|\|\mathbf{x}^k - \mathbf{x}^\star\|^2 = \widetilde{M}\|\mathbf{x}^k - \mathbf{x}^\star\|^2,$$

  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 [2].
- For more details on quasi-Newton methods, see Nocedal&Wright's book [7].

*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 [7] (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

$$\min_{\mathbf{H}} \|\mathbf{H} - \mathbf{H}_k\|_{\mathbf{W}} \quad \text{subject to } \mathbf{H} = \mathbf{H}^T \text{ and } \mathbf{H}\mathbf{y}^k = \mathbf{s}^k \tag{1}$$

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.

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.

Recall: Convergence bounds for non-convex problems

Lower bound

Consider $f \in \mathcal{F}_L^{1,1}$ and f is non-convex. Then any first-order method must satisfy,

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

As a corollary,

$$T = \Omega\left(\epsilon^{-2}\right) [1]$$

Convergence of adaptive methods for non-convex problems

- For convex problems, adaptive methods not always have proper convergence analysis.
- Similarly in non-convex setting, difficult to find a rigorous convergence bound.

Convergence of AdaGrad (non-convex)

Assume that $f \in \mathcal{F}_L^{1,1}$ and $f^\star = \min f(x) > \infty$. The scalar step-size version of AdaGrad satisfies:

$$\min_{k \in 1,..,T} \|\nabla f(x^k)\|^2 = O\left(\frac{(f(x_0) - f^\star)^2}{T}\right)$$

- This characterization of convergence is weaker than $\|\nabla f(x_T)\|^2 = O(1/T)$.

Recall: Logistic regression with non-convex regularizer

Problem (**Regularized logistic regression**)

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

$$f^\star := \min_{\mathbf{x},\beta} \left\{ f(\mathbf{x}) := \frac{1}{n} \sum_{j=1}^{n} \log\left(1 + \exp\left(-\mathbf{b}_j(\mathbf{a}_j^T \mathbf{x} + \beta)\right)\right) + \frac{\theta}{2}\phi(\mathbf{x}) \right\}.$$

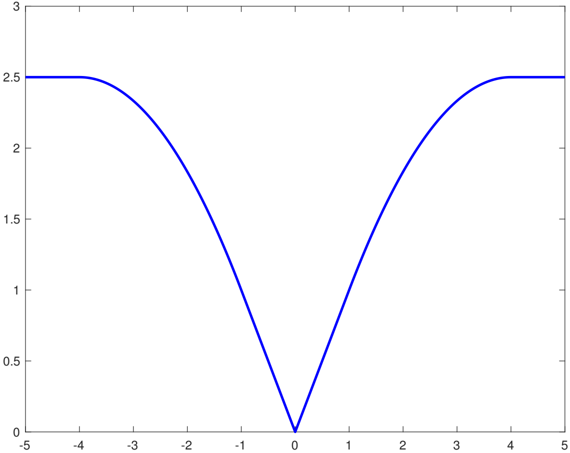
where $\phi(\mathbf{x}) = \sum_{i=1}^{d} \phi(\mathbf{x}_i)$.

Definition (Smoothly clipped absolute deviation (SCAD))

$$\phi(\mathbf{x}_i) = \begin{cases} \lambda\,|\mathbf{x}_i| & |\mathbf{x}_i| \le \lambda, \\ \left(-|\mathbf{x}_i|^2 + 2a\lambda\,|\mathbf{x}_i| - \lambda^2\right)/(2(a-1)) & \lambda < |\mathbf{x}_i| \le a\lambda, \\ (1+a)\lambda^2/2 & |\mathbf{x}_i| > a\lambda \end{cases}$$

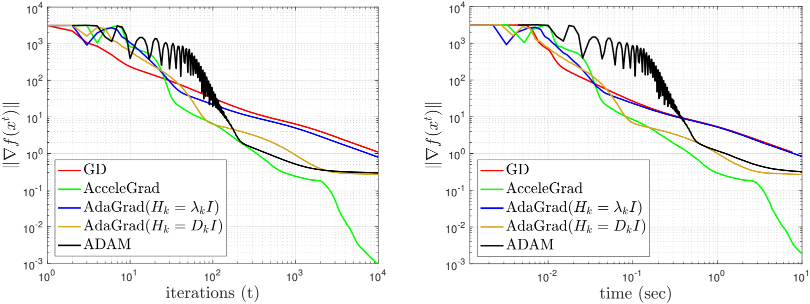
SCAD penalty

SCAD regularizer with $\lambda = 1$, $a = 4$.



Example: Convergence plot

Convergence and time plots of GD and adaptive methods for nonconvex logistic regression problem.



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
Lipschitz-gradient $f \in \mathcal{F}_L^{2,1}(\mathbb{R}^p)$	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
Strongly convex, smooth $f \in \mathcal{F}_{L,\mu}^{2,1}(\mathbb{R}^p)$	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
Lipschitz-gradient $f \in \mathcal{F}_L^{2,1}(\mathbb{R}^p)$	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
Strongly convex, smooth $f \in \mathcal{F}_{L,\mu}^{2,1}(\mathbb{R}^p)$	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
Lipschitz-gradient $f \in \mathcal{F}_L^{2,1}(\mathbb{R}^p)$	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
Strongly convex, smooth $f \in \mathcal{F}_{L,\mu}^{2,1}(\mathbb{R}^p)$	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

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

References I

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

[2] 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.

[3] 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.

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

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

References II

[6] 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.

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

[8] 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.