Mathematics of Data: From Theory to Computation

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 6: Unconstrained, smooth minimization III

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

EE-556 (Fall 2015)

lions@epfl

MARIE CURIE ACTIONS
FNSNF
FONDS NATIONAL SUISSE
SCHWEIZERISCHER NATIONALFONDS
FONDO NAZIONALE SVIZZERO
SWISS NATIONAL SCIENCE FOUNDATION
erc
EPFL

License Information for Mathematics of Data Slides

- This work is released under a Creative Commons License with the following terms:
- **Attribution**
  - The licensor permits others to copy, distribute, display, and perform the work. In return, licensees must give the original authors credit.
- **Non-Commercial**
  - The licensor permits others to copy, distribute, display, and perform the work. In return, licensees may not use the work for commercial purposes – unless they get the licensor's permission.
- **Share Alike**
  - The licensor permits others to distribute derivative works only under a license identical to the one that governs the licensor's work.
- Full Text of the License

Outline

- This lecture
  1. The quadratic case and conjugate gradient
  2. Other optimization methods
- Next lecture
  1. Motivation for non-smooth models
  2. Subgradient descent

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?

Linear systems

Problem (Solving a linear system)

Which is the best method for solving the linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b} \, ?$$

Solving a linear system via optimization

To find a solution to the linear system, we can also solve the optimization problem

$$\min_{\mathbf{x}} f_{\mathbf{A},\mathbf{b}}(\mathbf{x}) := \frac{1}{2}\langle \mathbf{A}\mathbf{x}, \mathbf{x}\rangle - \langle \mathbf{b}, \mathbf{x}\rangle$$

which is seen to have a solution satisfying $\mathbf{A}\mathbf{x} = \mathbf{b}$ by solving $\nabla_{\mathbf{x}} f_{\mathbf{A},\mathbf{b}}(\mathbf{x}) = 0$.

- $f_{\mathbf{A},\mathbf{b}}$ is a quadratic function with **Lipschitz-gradient** ($L = \|\mathbf{A}\|$).
- If $\mathbf{A}$ is a $p \times p$ symmetric positive definite matrix, (i.e., $\mathbf{A} = \mathbf{A}^T \succ 0$), $f_{\mathbf{A}}$ is also **strongly convex** ($\mu = \lambda_1(\mathbf{A})$, the smallest eigenvalue of $\mathbf{A}$).
- if $\mathbf{A}$ is not symmetric, but full column rank, we can consider the system

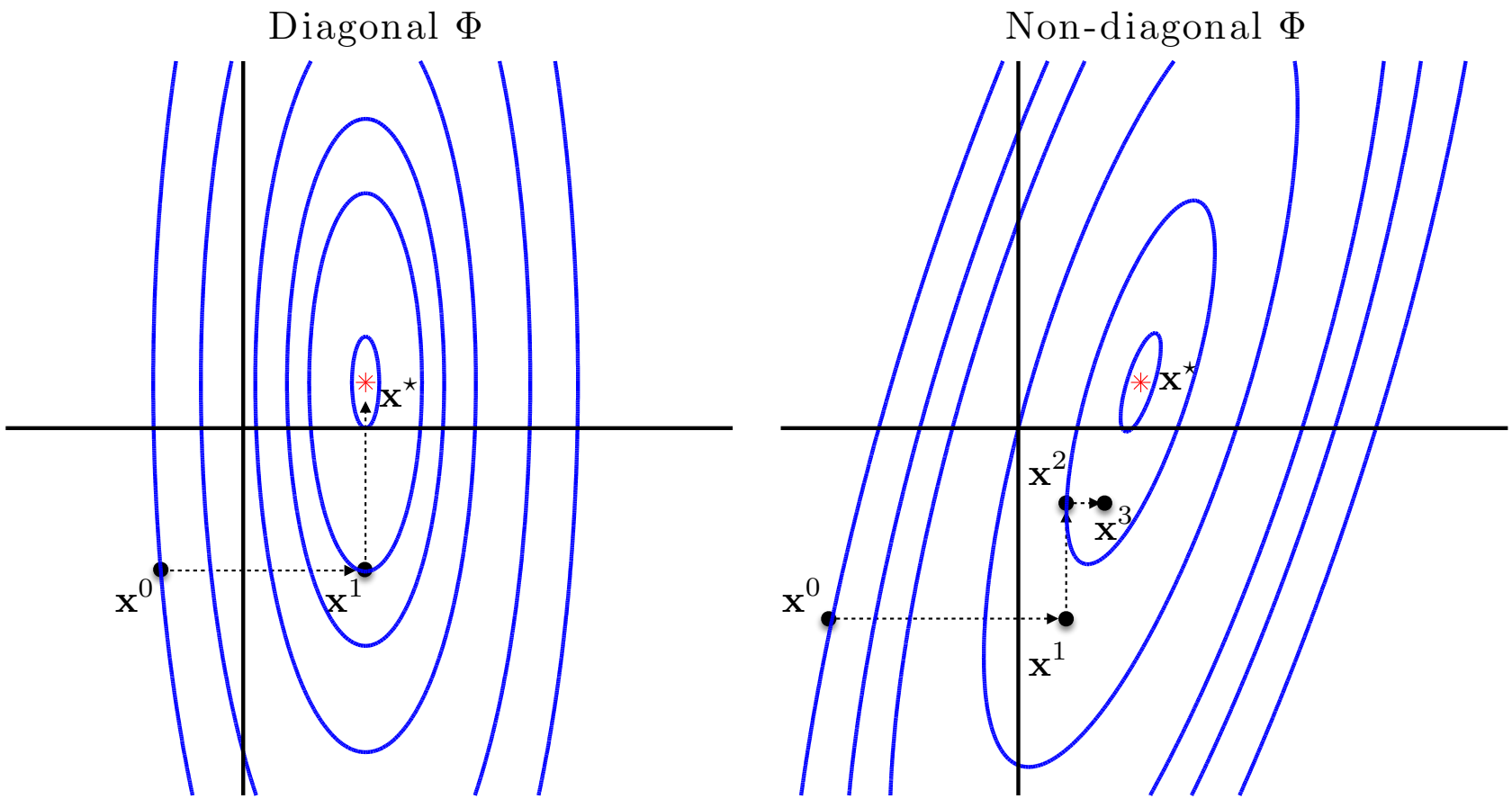
$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$$

which can be seen as: $\mathbf{\Phi}\mathbf{x} = \mathbf{y}$ where $\mathbf{\Phi}$ is symmetric and positive definite.

Linear systems

Remark

If $\mathbf{\Phi}$ is diagonal and positive definite, given a starting point $\mathbf{x}^0 \in \mathrm{dom}(f)$, successive minimization of $f_{\mathbf{\Phi},\mathbf{y}}(\mathbf{x})$ along the coordinate axes yield $\mathbf{x}^\star$ is at most p steps.



How can we adapt to the geometry of Φ?

Conjugate gradients method - Φ symmetric and positive definite

Generate a set of *conjugate* directions $\{\mathbf{p}^0, \mathbf{p}^1, \dots, \mathbf{p}^{p-1}\}$ such that

$$\langle \mathbf{p}^i, \mathbf{\Phi} \mathbf{p}^j \rangle = 0 \qquad \text{for all } i \neq j \qquad \text{(which also implies linear independence).}$$

Successively minimize $f_{\mathbf{\Phi},\mathbf{y}}$ along the individual conjugate directions. Let

$$\mathbf{r}^k = \mathbf{\Phi} \mathbf{x}^k - \mathbf{y} \quad \text{and} \quad \mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}^k \ ,$$

where α_k is the minimizer of $f_{\mathbf{\Phi},\mathbf{y}}(\mathbf{x})$ along $\mathbf{x}^k + \alpha \mathbf{p}^k$, i.e.,

$$\alpha_k = - \frac{\langle \mathbf{r}^k, \mathbf{p}^k \rangle}{\langle \mathbf{p}^k, \mathbf{\Phi} \mathbf{p}^k \rangle}$$

Theorem

For any $\mathbf{x}^0 \in \mathbb{R}^p$ the sequence $\{\mathbf{x}^k\}$ generated by the conjugate directions algorithm converges to the solution $\mathbf{x}^\star$ of the linear system in **at most** p *steps.*

Intuition

The conjugate directions adapt to the geometry of the problem, taking the role of the canonical directions when $\mathbf{\Phi}$ is a generic symmetric positive definite matrix.

Conjugate gradients method

Intuition

The conjugate directions adapt to the geometry of the problem, taking the role of the canonical directions when $\mathbf{\Phi}$ is a generic symmetric positive definite matrix.

Back to diagonal

For a generic symmetric positive definite $\mathbf{\Phi}$, let us consider the variable $\bar{\mathbf{x}} := \mathbf{S}^{-1}\mathbf{x}$, with

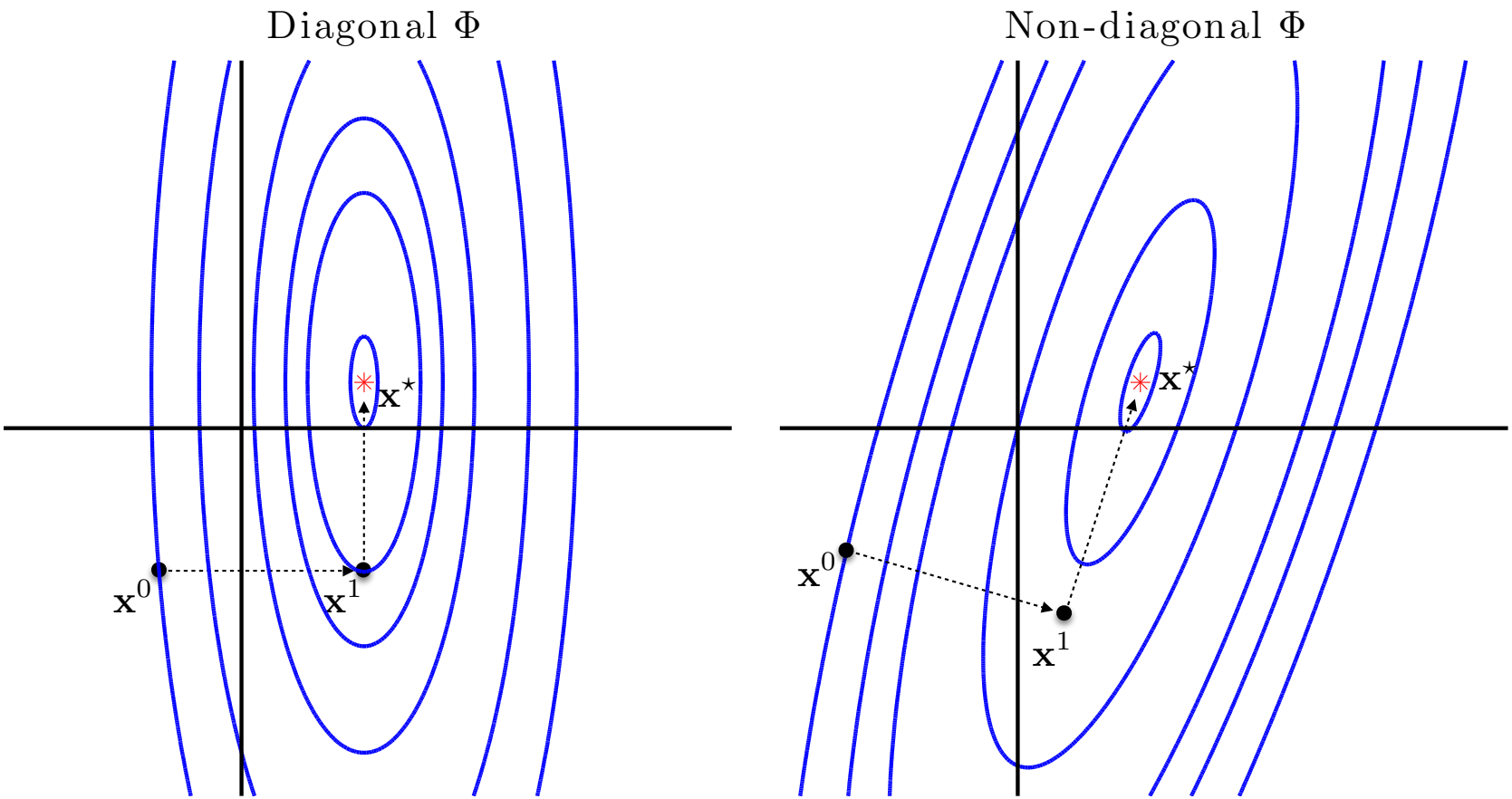
$$\mathbf{S} = \left[\mathbf{p}^0, \dots, \mathbf{p}^{p-1}\right]$$

where $\{\mathbf{p}^k\}$ are the conjugate directions with respect to $\mathbf{\Phi}$. $f_{\mathbf{\Phi},\mathbf{y}}(\mathbf{x})$ now becomes

$$\bar{f}_{\mathbf{\Phi},\mathbf{y}}(\bar{\mathbf{x}}) := f_{\mathbf{\Phi},\mathbf{y}}(\mathbf{S}\bar{\mathbf{x}}) = \frac{1}{2}\langle \bar{\mathbf{x}}, (\mathbf{S}^T\mathbf{\Phi}\mathbf{S})\bar{\mathbf{x}}\rangle - \langle \mathbf{S}^T\mathbf{y}, \bar{\mathbf{x}}\rangle.$$

By the conjugacy property, $\langle \mathbf{p}^i, \mathbf{\Phi}\mathbf{p}^j\rangle = 0, \ \forall\ i \neq j$, the matrix $\mathbf{S}^T\mathbf{\Phi}\mathbf{S}$ is diagonal. Therefore, we can find the minimum of $\bar{f}(\bar{\mathbf{x}})$ in at most p steps along the canonical directions in $\bar{\mathbf{x}}$ space, which are the $\{\mathbf{p}^k\}$ directions in $\mathbf{x}$ space.

Conjugate directions naturally adapt to the linear operator



Conjugate gradients method

Theorem

For any $\mathbf{x}^0 \in \mathbb{R}^p$ the sequence $\{\mathbf{x}^k\}$ generated by the conjugate directions algorithm converges to the solution $\mathbf{x}^\natural$ of the linear system in **at most** p *steps.*

Proof.

Since $\{\mathbf{p}^k\}$ are linearly independent, they span $\mathbb{R}^p$. Therefore, we can write

$$\mathbf{x}^\star - \mathbf{x}^0 = a_0\mathbf{p}^0 + a_1\mathbf{p}^1 + \cdots + a_{p-1}\mathbf{p}^{p-1}$$

for some values of the coefficients a_k. By multiplying on the left by $(\mathbf{p}^k)^T\mathbf{\Phi}$ and using the conjugacy property, we obtain

$$a_k = \frac{\langle \mathbf{p}^k, \mathbf{\Phi}(\mathbf{x}^\star - \mathbf{x}^0)\rangle}{\langle \mathbf{p}^k, \mathbf{\Phi}\mathbf{p}^k\rangle}.$$

Since $\mathbf{x}^k = \mathbf{x}^{k-1} + \alpha_{k-1}\mathbf{p}^{k-1}$, we have $\mathbf{x}^k = \mathbf{x}^0 + \alpha_0\mathbf{p}^0 + \alpha_1\mathbf{p}^1 + \cdots + \alpha_{k-1}\mathbf{p}^{k-1}$. By premultiplying by $(\mathbf{p}^k)^T\mathbf{\Phi}$ and using the conjugacy property, we obtain $\langle \mathbf{p}^k, \mathbf{\Phi}(\mathbf{x}^k - \mathbf{x}^0)\rangle = 0$ which implies

$$\langle \mathbf{p}^k, \mathbf{\Phi}(\mathbf{x}^\star - \mathbf{x}^0)\rangle = \langle \mathbf{p}^k, \mathbf{\Phi}(\mathbf{x}^\star - \mathbf{x}^k)\rangle = \langle \mathbf{p}^k, \mathbf{y} - \mathbf{\Phi}\mathbf{x}^0)\rangle = -\langle \mathbf{p}^k, \mathbf{r}^k\rangle$$

so that $a_k = -\frac{\langle \mathbf{p}^k, \mathbf{r}^k\rangle}{\langle \mathbf{p}^k, \mathbf{\Phi}\mathbf{p}^k\rangle} = \alpha_k$. □

Conjugate gradients method

How can we efficiently generate a set of conjugate directions?

Iteratively generate the new descent direction $\mathbf{p}^k$ from the previous one:

$$\mathbf{p}^k = -\mathbf{r}^k + \beta_k \mathbf{p}^{k-1}$$

For ensuring conjugacy $\langle \mathbf{p}^k, \mathbf{\Phi} \mathbf{p}^{k-1} \rangle = 0$, we need to choose β_k as

$$\beta_k = \frac{\langle \mathbf{r}^k, \mathbf{\Phi} \mathbf{p}^{k-1} \rangle}{\langle \mathbf{p}^{k-1}, \mathbf{\Phi} \mathbf{p}^{k-1} \rangle} \ .$$

Lemma

The directions $\{\mathbf{p}^0, \mathbf{p}^1, \ldots, \mathbf{p}^p\}$ *form a* conjugate directions *set.*

Conjugate gradients method

Conjugate gradients (CG) method

1 Initialization:

1.a Choose $\mathbf{x}^0 \in \mathrm{dom}(f)$ arbitrarily as a starting point.

1.b Set $\mathbf{r}^0 = \mathbf{\Phi x}^0 - \mathbf{y}$, $\mathbf{p}^0 = -\mathbf{r}^0$, $k = 0$.

2. While $\mathbf{r}^k \neq \mathbf{0}$, generate a sequence $\{\mathbf{x}^k\}_{k \geq 0}$ as:

$$
\begin{aligned}
\alpha_k &= -\frac{\langle \mathbf{r}^k, \mathbf{p}^k \rangle}{\langle \mathbf{p}^k, \mathbf{\Phi p}^k \rangle} \\
\mathbf{x}^{k+1} &= \mathbf{x}^k + \alpha_k \mathbf{p}^k \\
\mathbf{r}^{k+1} &= \mathbf{\Phi x}^{k+1} - \mathbf{y} \\
\beta_{k+1} &= \frac{\langle \mathbf{r}^{k+1}, \mathbf{\Phi p}^k \rangle}{\langle \mathbf{p}^k, \mathbf{\Phi p}^k \rangle} \\
\mathbf{p}^{k+1} &= -\mathbf{r}^{k+1} + \beta_{k+1} \mathbf{p}^k \\
k &= k + 1
\end{aligned}
$$

Theorem

Since the directions $\{\mathbf{p}^0, \mathbf{p}^1, \ldots, \mathbf{p}^k\}$ are conjugate, CG converges in at most p steps.

Other properties of the conjugate gradient method

Theorem

If $\mathbf{\Phi}$ has only r distinct eigenvalues, then the CG iterations will terminate at the solution in at most r iterations.

Theorem

If $\mathbf{\Phi}$ has eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_p$, we have that

$$\|\mathbf{x}^{k+1} - \mathbf{x}^\star\|_{\mathbf{\Phi}} \leq \left(\frac{\lambda_{p-k} - \lambda_1}{\lambda_{p-k} + \lambda_1} \right) \|\mathbf{x}^0 - \mathbf{x}^\star\|_{\mathbf{\Phi}},$$

where the local norm is given by $\|\mathbf{x}\|_{\mathbf{\Phi}} = \sqrt{\mathbf{x}^T \mathbf{\Phi} \mathbf{x}}$.

Theorem

Conjugate gradients algorithm satisfy the following iteration invariant for the solution of $\mathbf{\Phi}\mathbf{x} = \mathbf{y}$

$$\|\mathbf{x}^{k+1} - \mathbf{x}^\star\|_{\mathbf{\Phi}} \leq 2 \left(\frac{\sqrt{\kappa(\mathbf{\Phi})} - 1}{\sqrt{\kappa(\mathbf{\Phi})} + 1} \right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_{\mathbf{\Phi}},$$

where the condition number of $\mathbf{\Phi}$ is defined as $\kappa(\mathbf{\Phi}) := \|\mathbf{\Phi}\| \|\mathbf{\Phi}^{-1}\| = \frac{\lambda_p}{\lambda_1}$.

GD and AGD for the quadratic case: choice of the step size

Gradient Descent

$$\alpha_k = \frac{2}{L+\mu} \quad \text{with } L = \lambda_p(\mathbf{\Phi}) \text{ and } \mu = \lambda_1(\mathbf{\Phi})$$

Steepest descent

Choose α_k so as to minimize $f(\mathbf{x}^{k+1})$.

$$\alpha_k = \frac{\|\nabla f(\mathbf{x}^k)\|^2}{\langle \nabla f(\mathbf{x}^k), \mathbf{\Phi} \nabla f(\mathbf{x}^k) \rangle} \tag{1}$$

Barzilai-Borwein

$$\alpha_k = \frac{\|\nabla f(\mathbf{x}^{k-1})\|^2}{\langle \nabla f(\mathbf{x}^{k-1}), \mathbf{\Phi} \nabla f(\mathbf{x}^{k-1}) \rangle} \tag{2}$$

The quadratic case - convergence rates summary

Convergence rates

Gradient descent $\left(\alpha_k = \frac{2}{L+\mu}\right)$: $\|\mathbf{x}^k - \mathbf{x}^\star\|_2 \leq \left(\frac{\lambda_p - \lambda_1}{\lambda_p}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_2$

Steepest descent: $\|\mathbf{x}^{k+1} - \mathbf{x}^\star\|_{\mathbf{\Phi}} \leq \left(\frac{\lambda_p - \lambda_1}{\lambda_p + \lambda_1}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_{\mathbf{\Phi}}$

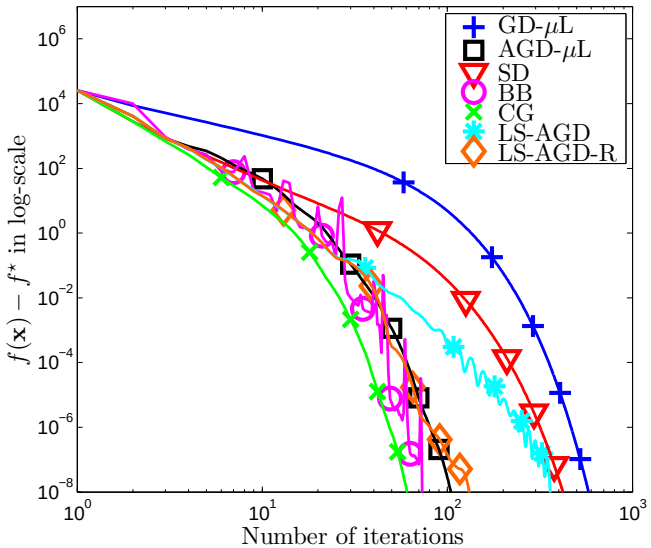
Barzilai-Borwein $(\lambda_p < 2\lambda_1)$: $\|\mathbf{x}^{k+1} - \mathbf{x}^\star\|_2 \leq \left(\frac{\lambda_p - \lambda_1}{\lambda_1}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_2$

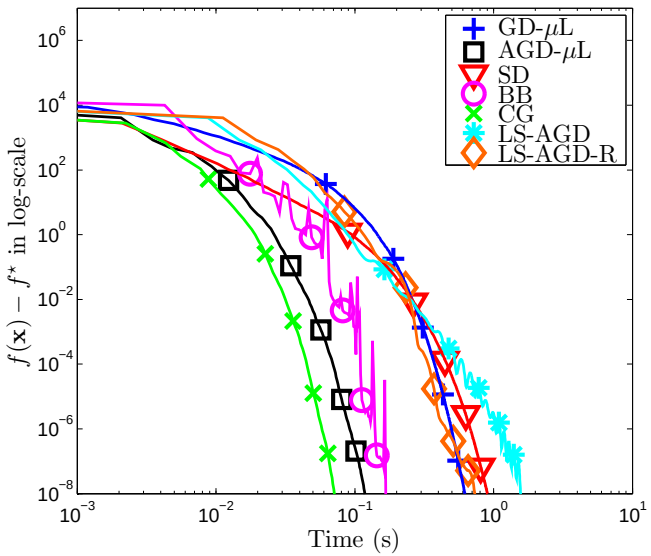
AGD-μL: $\|\mathbf{x}^k - \mathbf{x}^\star\|_2 \leq \left(\frac{\sqrt{\lambda_p} - \sqrt{\lambda_1}}{\sqrt{\lambda_p}}\right)^{\frac{k}{2}} \|\mathbf{x}^0 - \mathbf{x}^\star\|_2$

Conjugate gradient method: $\|\mathbf{x}^{k+1} - \mathbf{x}^\star\|_{\mathbf{\Phi}} \leq \left(\frac{\sqrt{\lambda_p} - \sqrt{\lambda_1}}{\sqrt{\lambda_p} + \sqrt{\lambda_1}}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_{\mathbf{\Phi}}$

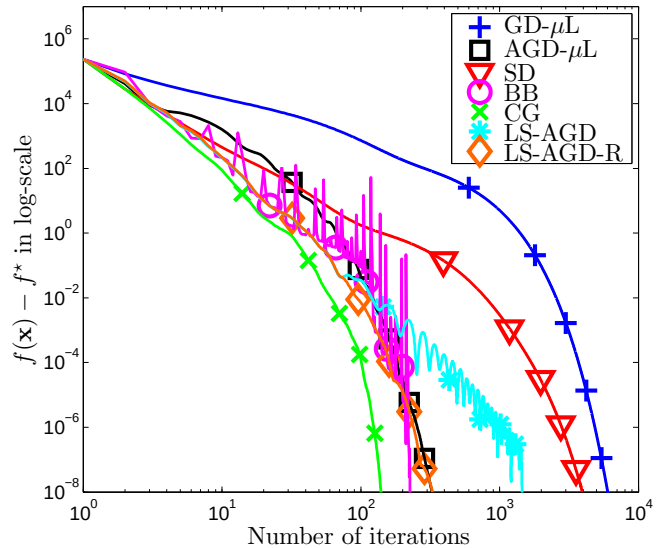
Example: Quadratic function

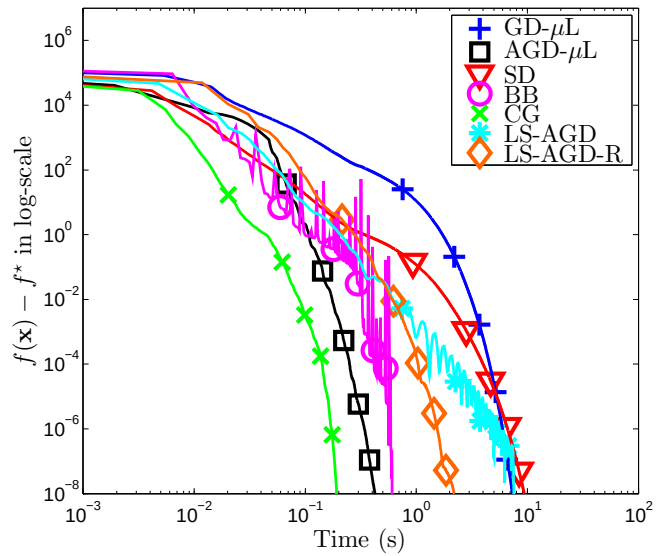
Case 1: $n = p = 1000, \kappa(\mathbf{A}) = 100$



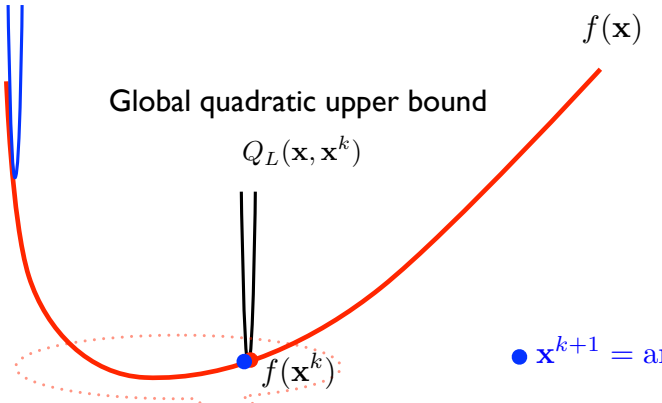


Case 2: $n = p = 1000, \kappa(\mathbf{A}) = 1000$





How can we better adapt to the local geometry?

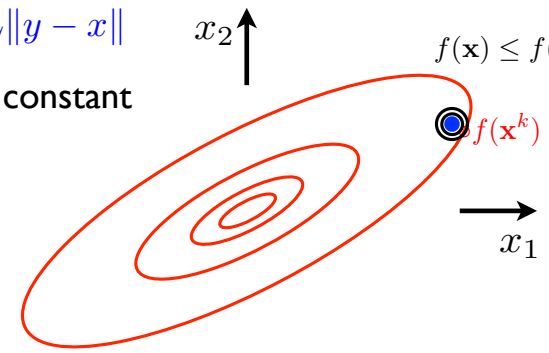


- $\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L}{2} \|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$

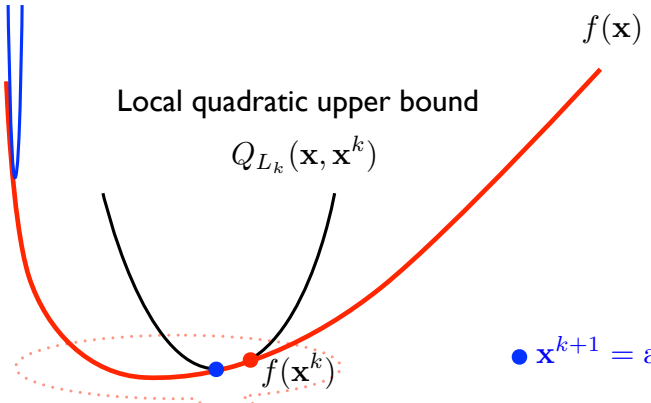
$\|\nabla f(x) - \nabla f(y)\| \leq L \|y - x\|$

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?



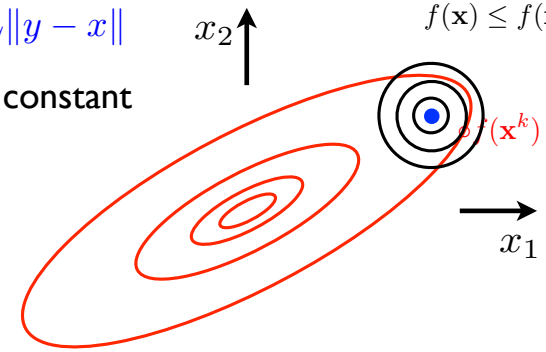
- $\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L_k}{2} \|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$

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

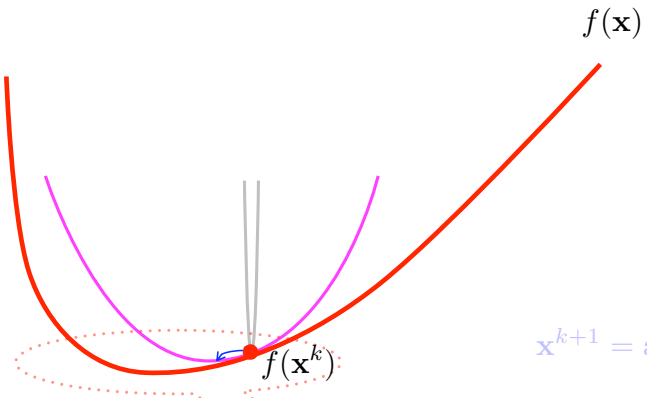
L is a global worst-case constant

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

applies only locally



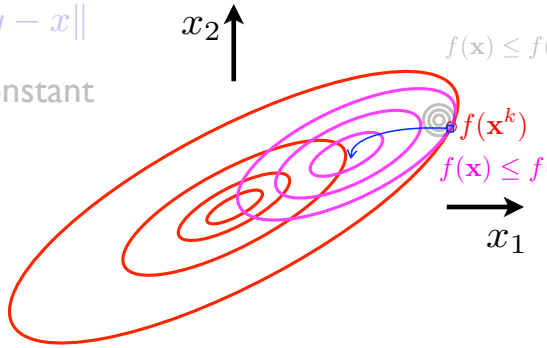
How can we better adapt to the local geometry?



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

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

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.

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 $\mathbf{H}_k$

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

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* $\mathbf{x}, \mathbf{y} \in dom(f)$.

Moreover, assume the starting point $\mathbf{x}^0 \in dom(f)$ *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 [2]

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

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

By Taylor's theorem, we also have

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

Combining the two above, we obtain

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

Locally quadratic convergence of the Newton method–II

Newton's method local quadratic convergence - Proof [2].

- 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

Problem (Logistic regression)

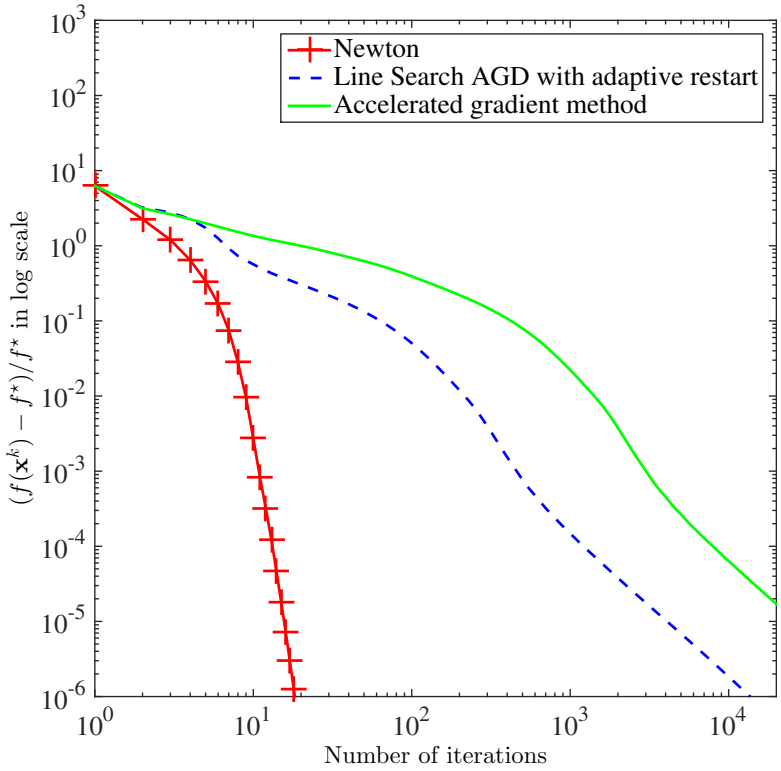
Given $\mathbf{A} \in \{0,1\}^{n \times p}$ *and* $\mathbf{b} \in \{-1,+1\}^n$*, solve:*

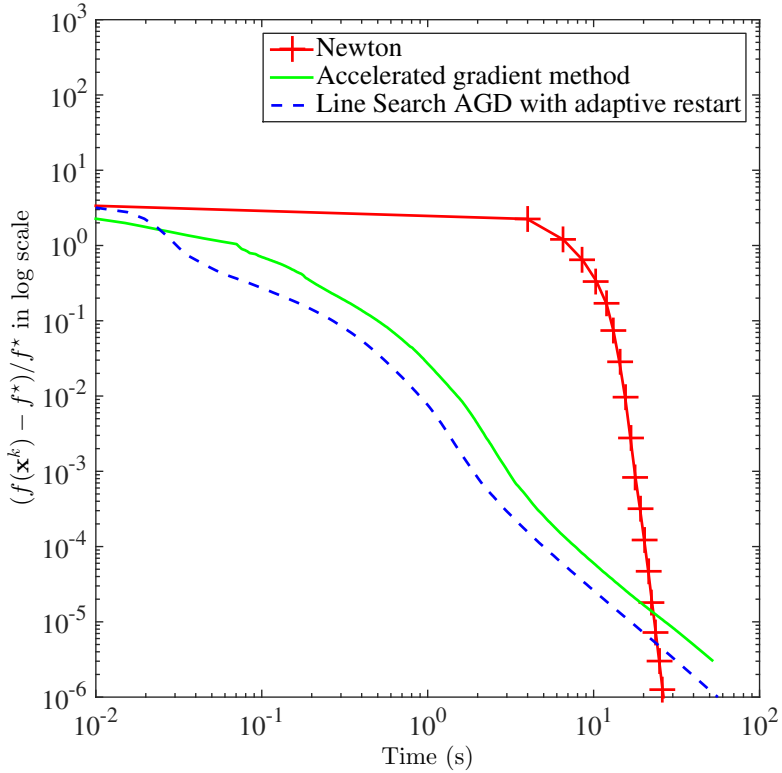
$$f^\star := \min_{\mathbf{x},\beta} \left\{ f(\mathbf{x}) := \frac{1}{n} \sum_{j=1}^{n} \log\left(1 + \exp\left(-\mathbf{b}_j(\mathbf{a}_j^T \mathbf{x} + \beta)\right)\right) \right\}.$$

Real data

- Real data: `w5a` with $n = 9888$ data points, $p = 300$ features
- Available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`.

Example: Logistic regression - numerical results





Parameters

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

Quasi-Newton methods

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

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 $n \gg p$.

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:

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

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).

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 $c_1 = 0.1$.

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 [1].

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 [1].
- For more details on quasi-Newton methods, see Nocedal&Wright's book [2].

*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 [2] (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

$$\min_{\mathbf{H}} \|\mathbf{H} - \mathbf{H}_k\|_{\mathbf{W}} \quad \text{subject to } \mathbf{H} = \mathbf{H}^T \text{ and } \mathbf{H}\mathbf{y}^k = \mathbf{s}^k \tag{3}$$

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

- 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 [2] (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

$$\min_{\mathbf{H}} \|\mathbf{H} - \mathbf{H}_k\|_{\mathbf{W}} \quad \text{subject to } \mathbf{H} = \mathbf{H}^T \text{ and } \mathbf{H}\mathbf{y}^k = \mathbf{s}^k \tag{3}$$

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

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 of 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 $i = k-1, \dots, k-m$

$$\begin{aligned} \alpha_i &= \eta_i \langle \mathbf{s}^i, \mathbf{q} \rangle \\ \mathbf{q} &= \mathbf{q} - \alpha_i \mathbf{y}^i. \end{aligned}$$

3. $\mathbf{r} = \mathbf{H}_k^0 \mathbf{q}$.
4. For $i = k-m, \dots, k-1$

$$\begin{aligned} \beta &= \eta_i \langle \mathbf{y}^i, \mathbf{r} \rangle \\ \mathbf{r} &= \mathbf{r} + (\alpha_i - \beta) \mathbf{s}^i. \end{aligned}$$

5. $\mathbf{H}_k \nabla f(\mathbf{x}^k) = \mathbf{r}$.

Remarks

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

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

L-BFGS: A quasi-Newton method

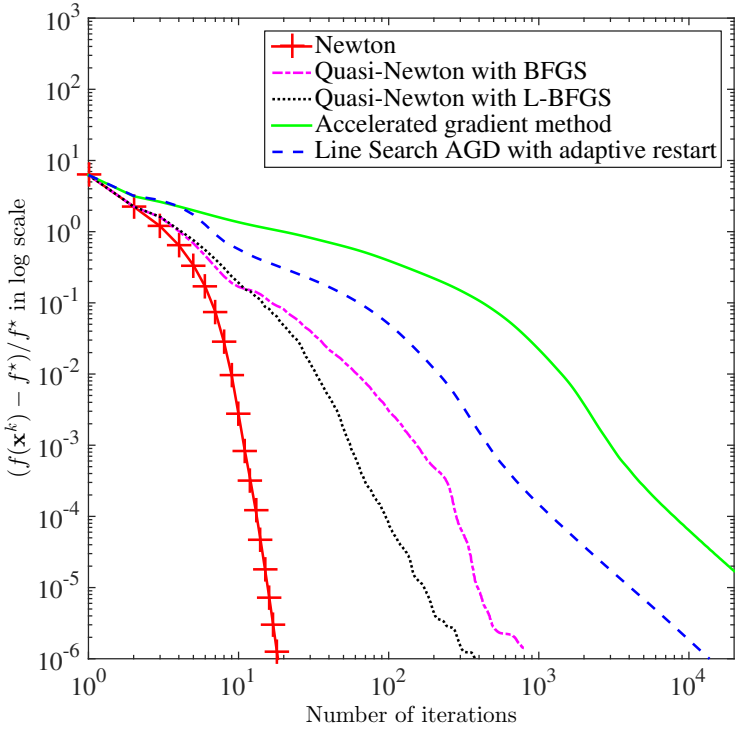
L-BFGS

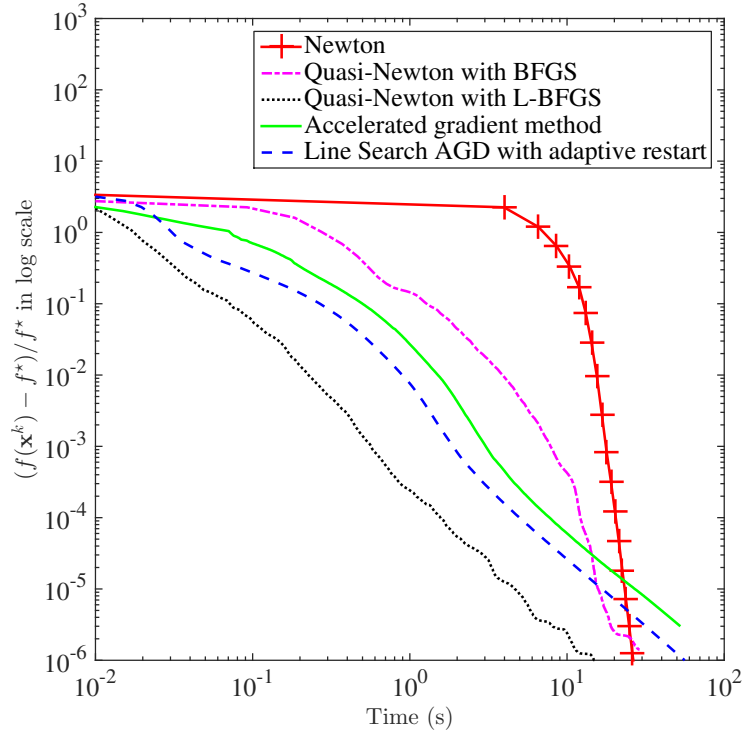
1. Choose starting point $\mathbf{x}^0$ and $m > 0$.
2. For $k = 0, 1, \ldots$
   - **2.a** Choose $\mathbf{H}_k^0$.
   - **2.b** Compute $\mathbf{p}^k = -\mathbf{H}_k \nabla f(\mathbf{x}^k)$ using the previous algorithm.
   - **2.c** Set $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}^k$, where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

Warning

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

Example: Logistic regression - numerical results





Parameters

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

Performance of optimization algorithms

Time-to-reach ϵ

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

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

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

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

Assumptions on f	Algorithm	Convergence rate	Iteration complexity
Lipschitz-gradient $f \in \mathcal{F}_L^{2,1}(\mathbb{R}^p)$	Gradient descent	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
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
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Linear (e^{-k}), Quadratic	One gradient, one linear system

Performance of optimization algorithms

A non-exhaustive comparison:

Assumptions on f	Algorithm	Convergence rate	Iteration complexity
Lipschitz-gradient $f \in \mathcal{F}_L^{2,1}(\mathbb{R}^p)$	Gradient descent	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
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
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	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}.

Performance of optimization algorithms

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
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
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
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Linear (e^{-k}), Quadratic	One gradient, one linear system

Main computations of the Quasi-Newton method, which we will discuss in the sequel

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

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

Performance of optimization algorithms

A non-exhaustive comparison:

Assumptions on f	Algorithm	Convergence rate	Iteration complexity
Lipschitz-gradient $f \in \mathcal{F}_L^{2,1}(\mathbb{R}^p)$	Gradient descent	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
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
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Linear (e^{-k}), Quadratic	One gradient, one linear system

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

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

References I

[1] 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.

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