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 2018)
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. Stochastic gradient methods
Recommended reading

Motivation

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

Problem (Mathematical formulation)

\[ F^* := \min_{x \in \mathbb{R}^p} \{ F(x) := f(x) \} \]  \hspace{1cm} (1)

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

How do 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

\[ Ax = b \]?

Solving a linear system via optimization

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

\[
\min_x f_{A,b}(x) := \frac{1}{2} \langle Ax, x \rangle - \langle b, x \rangle
\]

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

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

\[
A^T Ax = A^T b
\]

which can be seen as: \( \Phi x = y \) where \( \Phi \) is symmetric and positive definite.
Linear systems

Remark

If $\Phi$ is diagonal and positive definite, given a starting point $x^0 \in \text{dom}(f)$, successive minimization of $f_{\Phi,y}(x)$ along the coordinate axes yield $x^*$ is at most $p$ steps.
How can we adapt to the geometry of $\Phi$?

**Conjugate gradients method - $\Phi$ symmetric and positive definite**

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

$$\langle p^i, \Phi p^j \rangle = 0 \quad \text{for all } i \neq j$$

(which also implies linear independence).

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

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

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

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

**Theorem**

*For any $x^0 \in \mathbb{R}^p$ the sequence $\{x^k\}$ generated by the conjugate directions algorithm converges to the solution $x^*$ 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 $\Phi$ 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 $\Phi$ is a generic symmetric positive definite matrix.

**Back to diagonal**

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

$$S = \begin{bmatrix} p^0, \ldots, p^{p-1} \end{bmatrix}$$

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

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

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

Diagonal $\Phi$

Non-diagonal $\Phi$
Conjugate gradients method

Theorem

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

Proof.

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

\[
x^* - x^0 = a_0 p^0 + a_1 p^1 + \cdots + a_{p-1} p^{p-1}
\]

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

\[
a_k = \frac{\langle p^k, \Phi(x^* - x^0) \rangle}{\langle p^k, \Phi p^k \rangle}.
\]

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

\[
\langle p^k, \Phi(x^k - x^0) \rangle = 0
\]

which implies

\[
\langle p^k, \Phi(x^* - x^0) \rangle = \langle p^k, \Phi(x^* - x^k) \rangle = \langle p^k, y - \Phi x^k \rangle = -\langle p^k, r^k \rangle
\]

so that \( a_k = -\frac{\langle p^k, x^k \rangle}{\langle p^k, \Phi p^k \rangle} = \alpha_k \). \( \square \)
Conjugate gradients method

How can we efficiently generate a set of conjugate directions?

Iteratively generate the new descent direction $p^k$ from the previous one:

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

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

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

Lemma

The directions $\{p^0, p^1, \ldots, p^p\}$ form a conjugate directions set.
## Conjugate gradients method

<table>
<thead>
<tr>
<th>Initialization:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.a Choose $x^0 \in \text{dom}(f)$ arbitrarily as a starting point.</td>
</tr>
<tr>
<td>1.b Set $r^0 = \Phi x^0 - y$, $p^0 = -r^0$, $k = 0$.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>While $r^k \neq 0$, generate a sequence ${x^k}_{k \geq 0}$ as:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_k = -\frac{\langle r^k, p^k \rangle}{\langle p^k, \Phi p^k \rangle}$</td>
</tr>
<tr>
<td>$x^{k+1} = x^k + \alpha_k p^k$</td>
</tr>
<tr>
<td>$r^{k+1} = \Phi x^{k+1} - y$</td>
</tr>
<tr>
<td>$\beta_{k+1} = \frac{\langle r^{k+1}, \Phi p^k \rangle}{\langle p^k, \Phi p^k \rangle}$</td>
</tr>
<tr>
<td>$p^{k+1} = -r^{k+1} + \beta_{k+1} p^k$</td>
</tr>
<tr>
<td>$k = k + 1$</td>
</tr>
</tbody>
</table>

### Theorem

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

---

*Mathematics of Data*  
Prof. Volkan Cevher, volkan.cevher@epfl.ch
Other properties of the conjugate gradient method

**Theorem**

*If* \( \Phi \) *has only* \( r \) *distinct eigenvalues, then the CG iterations will terminate at the solution in at most* \( r \) *iterations.*

**Theorem**

*If* \( \Phi \) *has eigenvalues* \( \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_p \), *we have that*

\[
\| x^{k+1} - x^* \|_\Phi \leq \left( \frac{\lambda_p - k - \lambda_1}{\lambda_p - k + \lambda_1} \right) \| x^0 - x^* \|_\Phi,
\]

*where the local norm is given by* \( \| x \|_\Phi = \sqrt{x^T \Phi x} \).

**Theorem**

*Conjugate gradients algorithm satisfy the following iteration invariant for the solution of* \( \Phi x = y \)

\[
\| x^{k+1} - x^* \|_\Phi \leq 2 \left( \frac{\sqrt{\kappa(\Phi)} - 1}{\sqrt{\kappa(\Phi)} + 1} \right)^k \| x^0 - x^* \|_\Phi,
\]

*where the condition number of* \( \Phi \) *is defined as* \( \kappa(\Phi) := \| \Phi \| \| \Phi^{-1} \| = \frac{\lambda_p}{\lambda_1} \).
GD and AGD for the quadratic case: choice of the step size

**Gradient Descent**

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

**Steepest descent**

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

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

**Barzilai-Borwein**

\[ \alpha_k = \frac{\| \nabla f(x^{k-1}) \|^2}{\langle \nabla f(x^{k-1}), \Phi \nabla f(x^{k-1}) \rangle} \tag{2} \]
### Convergence rates summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Convergence Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradient descent (( \alpha_k = \frac{2}{L+\mu} ))</td>
<td>[ | x^k - x^* |_2 \leq \left( \frac{\lambda_p - \lambda_1}{\lambda_p} \right)^k | x^0 - x^* |_2 ]</td>
</tr>
<tr>
<td>Steepest descent:</td>
<td>[ | x^{k+1} - x^* |<em>\Phi \leq \left( \frac{\lambda_p - \lambda_1}{\lambda_p + \lambda_1} \right)^k | x^0 - x^* |</em>\Phi ]</td>
</tr>
<tr>
<td>Barzilai-Borwein (( \lambda_p &lt; 2\lambda_1 ))</td>
<td>[ | x^{k+1} - x^* |_2 \leq \left( \frac{\lambda_p - \lambda_1}{\lambda_1} \right)^k | x^0 - x^* |_2 ]</td>
</tr>
<tr>
<td>AGD-(\mu)L:</td>
<td>[ | x^k - x^* |_2 \leq \left( \frac{\sqrt{\lambda_p} - \sqrt{\lambda_1}}{\sqrt{\lambda_p}} \right)^{\frac{k}{2}} | x^0 - x^* |_2 ]</td>
</tr>
<tr>
<td>Conjugate gradient method:</td>
<td>[ | x^{k+1} - x^* |<em>\Phi \leq \left( \frac{\sqrt{\lambda_p} - \sqrt{\lambda_1}}{\sqrt{\lambda_p} + \sqrt{\lambda_1}} \right)^k | x^0 - x^* |</em>\Phi ]</td>
</tr>
</tbody>
</table>
Example: Quadratic function

**Case 1:** \( n = p = 1000, \kappa(A) = 100 \)

**Case 2:** \( n = p = 1000, \kappa(A) = 1000 \)
How can we better adapt to the local geometry?

\[ f(x) \]

Global quadratic upper bound

\[ Q_L(x, x^k) \]

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

\[ \| \nabla f(x) - \nabla f(y) \| \leq L \| y - x \| \]

L is a global worst-case constant

\[ f(x) \leq f(x^k) + \nabla f(x^k)^T (x - x^k) + \frac{L}{2} \| x - x^k \|_2^2 \]
How can we better adapt to the local geometry?

\[ f(x) \]

Local quadratic upper bound

\[ Q_{L_k}(x, x^k) \]

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

\[ \| \nabla f(x) - \nabla f(y) \| \leq L \| y - x \| \]

L is a global worst-case constant

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

applies only locally
How can we better adapt to the local geometry?

\[ f(x) \]

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

\[ \| \nabla f(x) - \nabla f(y) \| \leq L \| y - x \| \]

L is a global worst-case constant

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

\[ f(x) \leq f(x^k) + \nabla f(x^k)^T (x - x^k) + \frac{1}{2} \| x - x^k \|_{H^{-1}}^2 \]

Mathematics of Data | Prof. Volkan Cevher, volkan.cevher@epfl.ch
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:

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

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

1. Choose $x^0 \in \mathbb{R}^p$ as a starting point and $H_0 \succ 0$.
2. For $k = 0, 1, \cdots$, perform:
   \[
   \begin{align*}
   d_k &:= -H_k^{-1}\nabla f(x^k), \\
   x^{k+1} &:= x^k + \alpha_k d^k,
   \end{align*}
   \]
   where $\alpha_k \in (0, 1]$ is a given step size.
3. Update $H_{k+1} \succ 0$ if necessary.

Common choices of the variable metric $H_k$

- $H_k := \lambda_k I$ $\implies$ gradient descent method.
- $H_k := D_k$ (a positive diagonal matrix) $\implies$ scaled gradient descent method.
- $H_k := \nabla^2 f(x^k)$ $\implies$ Newton method.
- $H_k \approx \nabla^2 f(x^k)$ $\implies$ quasi-Newton method.
Newton method

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

Local quadratic approximation using the Hessian

\[ f(x_k + p) \approx f(x_k) + \langle p, \nabla f(x_k) \rangle + \frac{1}{2} \langle p, \nabla^2 f(x_k) p \rangle \]

The Newton direction is the vector \( p_k \) that minimizes \( f(x_k + p) \); assuming the Hessian \( \nabla^2 f_k \) to be positive definite:

\[ \nabla^2 f_k p_k = -\nabla f(x_k) \iff p_k = -\left( \nabla^2 f_k \right)^{-1} \nabla f(x_k) \]

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

\[ x_{k+1} = x_k - \left( \nabla^2 f_k \right)^{-1} \nabla f(x_k) \]
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 $f(x^k + p)$:

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

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

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

  $$\nabla^2 f(x^k) p^k = -\nabla f(x^k) \iff p^k = -\left(\nabla^2 f(x^k)\right)^{-1} \nabla f(x^k)$$
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 $f(x^k + p)$:

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

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

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

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

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

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

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

\[
\nabla^2 f(x^k) p^k = -\nabla f(x^k) \iff p^k = -\left( \nabla^2 f(x^k) \right)^{-1} \nabla f(x^k)
\]

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

\[
x^{k+1} = x^k - \left( \nabla^2 f(x^k) \right)^{-1} \nabla f(x^k)
\]

Remark

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

Assume $f$ is a twice differentiable convex function with minimum at $x^*$ such that:

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

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

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

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, $\nabla^2 f(x^k)p^k = -\nabla f(x^k)$. 
Locally quadratic convergence of the Newton method–I

Newton’s method local quadratic convergence - Proof [?]

Since \( \nabla f(x^*) = 0 \) we have

\[
x^{k+1} - x^* = x^k - x^* - \left( \nabla^2 f(x^k) \right)^{-1} \nabla f(x^k)
\]

\[
= \left( \nabla^2 f(x^k) \right)^{-1} \left( \nabla^2 f(x^k)(x^k - x^*) - (\nabla f(x^k) - \nabla f(x^*)) \right)
\]

By Taylor’s theorem, we also have

\[
\nabla f(x^k) - \nabla f(x^*) = \int_0^1 \nabla^2 f(x^k + t(x^* - x^k))(x^k - x^*)dt
\]

Combining the two above, we obtain

\[
\| \nabla^2 f(x^k)(x^k - x^*) - (\nabla f(x^k) - \nabla f(x^*)) \|
\]

\[
= \left\| \int_0^1 \left( \nabla^2 f(x^k) - \nabla^2 f(x^k + t(x^* - x^k)) \right)(x^k - x^*)dt \right\|
\]

\[
\leq \int_0^1 \left\| \nabla^2 f(x^k) - \nabla^2 f(x^k + t(x^* - x^k)) \right\| \|x^k - x^*\|dt
\]

\[
\leq M \|x^k - x^*\|^2 \int_0^1 tdt = \frac{1}{2} M \|x^k - x^*\|^2
\]
Newton’s method local quadratic convergence - Proof [?].

- Recall

\[ x^{k+1} - x^* = (\nabla^2 f(x^k))^{-1} \left( \nabla^2 f(x^k)(x^k - x^*) - (\nabla f(x^k) - \nabla f(x^*)) \right) \]

\[ \| \nabla^2 f(x^k)(x^k - x^*) - (\nabla f(x^k) - \nabla f(x^*)) \| \leq \frac{1}{2} M \| x^k - x^* \|^2 \]

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

- Substituting, we obtain

\[ \| x^{k+1} - x^* \| \leq M \| (\nabla^2 f(x^*))^{-1} \| \| x^k - x^* \|^2 = \tilde{M} \| x^k - x^* \|^2, \]

where \( \tilde{M} = M \| (\nabla^2 f(x^*))^{-1} \| \).

- If we choose \( \| x^0 - x^* \| \leq \min(r, 1/(2\tilde{M})) \), we obtain by induction that the iterates \( x^k \) converge quadratically to \( x^* \).
Example: Logistic regression

Problem (Logistic regression)

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

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

Real data

- Real data: w5a with $n = 9888$ data points, $p = 300$ features
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 $x^*$ and $f^*$ by applying Newton’s method for 200 iterations.
Quasi-Newton methods

Quasi-Newton methods use an approximate Hessian oracle and can be more scalable.
▶ Useful for $f(x) := \sum_{i=1}^{n} f_i(x)$ with $n \gg p$.

Main ingredients

Quasi-Newton direction:

$$p^k = -H_k^{-1} \nabla f(x^k) = -B_k \nabla f(x^k).$$

▶ Matrix $H_k$, or its inverse $B_k$, 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 $\alpha_k$ is chosen to satisfy the Wolfe conditions:

$$f(x^k + \alpha_k p^k) \leq f(x^k) + c_1 \alpha_k \langle \nabla f(x^k), p^k \rangle \quad \text{(sufficient decrease)}$$

$$\langle \nabla f(x^k + \alpha_k p^k), p^k \rangle \geq c_2 \langle \nabla f(x^k), p^k \rangle \quad \text{(curvature condition)}$$

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 [?].

▶ For more details on quasi-Newton methods, see Nocedal&Wright’s book [?].
How do we update $B_{k+1}$?

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

$$m_{k+1}(\tilde{p}) := f(x^{k+1}) + \langle \nabla f(x^{k+1}), \tilde{p} - x^{k+1} \rangle + \frac{1}{2} \langle B_{k+1}(\tilde{p} - x^{k+1}), (\tilde{p} - x^{k+1}) \rangle.$$

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

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

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

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

$$\min_{H} \|H - H_k\|_W \quad \text{subject to } H = H^T \text{ and } Hy^k = s^k$$

The solution is a rank-2 update of the matrix $H_k$:

$$H_{k+1} = V_k^T H_k V_k + \eta_k s^k (s^k)^T,$$

where $V_k = I - \eta_k y^k (s^k)^T$.

- Initialization of $H_0$ is an art. We can choose to set it to be an approximation of $\nabla^2 f(x^0)$ obtained by finite differences or just a multiple of the identity matrix.
*Quasi-Newton methods*

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

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

$$\min_H \|H - H_k\|_W \quad \text{subject to } H = H^T \text{ and } H y^k = s^k \quad (3)$$

The solution is a rank-2 update of the matrix $H_k$:

$$H_{k+1} = V_k^T H_k V_k + \eta_k s^k (s^k)^T,$$

where $V_k = I - \eta_k y^k (s^k)^T$.

**Theorem (Convergence of BFGS)**

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

Then $x^k$ converges to $x^*$ at a superlinear rate.

**Remarks**

The proof shows that given the assumptions, the BFGS updates for $B_k$ 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 $H_k$.
- When $p$ is very large, $H_k$ can prohibitively expensive to store and apply.

**L(imited memory)-BFGS**

- Do not store $H_k$, but keep only the $m$ most recent pairs $\{(s^i, y^i)\}$.
- Compute $H_k \nabla f(x_k)$ by performing a sequence of operations with $s^i$ and $y^i$:
  - Choose a temporary initial approximation $H_k^0$.
  - Recursively apply $H_{k+1} = V_k^T H_k V_k + \eta_k s^k (s^k)^T$, $m$ times starting from $H_k^0$:
    \[
    H_k = \left( V_{k-1}^T \cdots V_{k-m}^T \right) H_k^0 \left( V_{k-m} \cdots V_{k-1} \right) \\
    + \eta_{k-m} \left( V_{k-1}^T \cdots V_{k-m+1}^T \right) s^{k-m} (s^{k-m})^T \left( V_{k-m+1} \cdots V_{k-1} \right) \\
    + \cdots \\
    + \eta_{k-1} s^{k-1} (s^{k-1})^T
    \]
  - From the previous expression, we can compute $H_k \nabla f(x^k)$ recursively.
- Replace the oldest element in $\{s^i, y^i\}$ with $(s^k, y^k)$.
- From practical experience, $m \in (3, 50)$ does the trick.
L-BFGS: A quasi-Newton method

**Procedure for computing** $H_k \nabla f(x^k)$

0. Recall $\eta_k = 1/\langle y^k, s^k \rangle$.
1. $q = \nabla f(x^k)$.
2. For $i = k - 1, \ldots, k - m$
   $\alpha_i = \eta_i \langle s^i, q \rangle$
   $q = q - \alpha_i y^i$.
3. $r = H_0^k q$.
4. For $i = k - m, \ldots, k - 1$
   $\beta = \eta_i \langle y^i, r \rangle$
   $r = r + (\alpha_i - \beta) s^i$.
5. $H_k \nabla f(x^k) = r$.

**Remarks**

- Apart from the step $r = H_0^k q$, the algorithm requires only $4mp$ multiplications.
- If $H_0^k$ is chosen to be diagonal, another $p$ multiplications are needed.
- An effective initial choice is $H_0^k = \gamma_k I$, where
  
  $\gamma_k = \frac{\langle s^{k-1}, y^{k-1} \rangle}{\langle y^{k-1}, y^{k-1} \rangle}$
L-BFGS: A quasi-Newton method

### L-BFGS

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

**Warning**

L-BFGS updates does not guarantee positive semidefiniteness of the variable metric $H_k$ 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 $x^*$ and $f^*$ by applying Newton’s method for 200 iterations.
Time-to-reach $\epsilon$

time-to-reach $\epsilon = \text{number of iterations to reach } \epsilon \times \text{per iteration time}$

The speed of numerical solutions depends on two factors:

- **Convergence rate** determines the number of iterations needed to obtain an $\epsilon$-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:

<table>
<thead>
<tr>
<th>Assumptions on $f$</th>
<th>Algorithm</th>
<th>Convergence rate</th>
<th>Iteration complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lipschitz-gradient</td>
<td>Gradient descent</td>
<td>Sublinear ($1/k$)</td>
<td>One gradient</td>
</tr>
<tr>
<td>$f \in F_{L}^{2,1}(\mathbb{R}^p)$</td>
<td>Accelerated GD</td>
<td>Sublinear ($1/k^2$)</td>
<td>One gradient</td>
</tr>
<tr>
<td></td>
<td>Quasi-Newton</td>
<td>Superlinear</td>
<td>One gradient, rank-2 update</td>
</tr>
<tr>
<td></td>
<td>Newton method</td>
<td>Sublinear ($1/k$), Quadratic</td>
<td>One gradient, one linear system</td>
</tr>
<tr>
<td>Strongly convex, smooth</td>
<td>Gradient descent</td>
<td>Linear ($e^{-k}$)</td>
<td>One gradient</td>
</tr>
<tr>
<td>$f \in F_{L,\mu}^{2,1}(\mathbb{R}^p)$</td>
<td>Accelerated GD</td>
<td>Linear ($e^{-k}$)</td>
<td>One gradient</td>
</tr>
<tr>
<td></td>
<td>Quasi-Newton</td>
<td>Superlinear</td>
<td>One gradient, rank-2 update</td>
</tr>
<tr>
<td></td>
<td>Newton method</td>
<td>Linear ($e^{-k}$), Quadratic</td>
<td>One gradient, one linear system</td>
</tr>
</tbody>
</table>
Performance of optimization algorithms

A non-exhaustive comparison:

<table>
<thead>
<tr>
<th>Assumptions on $f$</th>
<th>Algorithm</th>
<th>Convergence rate</th>
<th>Iteration complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lipschitz-gradient $f \in \mathcal{F}_{L,1}^2(\mathbb{R}^P)$</td>
<td>Gradient descent</td>
<td>Sublinear ($1/k$)</td>
<td>One gradient</td>
</tr>
<tr>
<td></td>
<td>Accelerated GD</td>
<td>Sublinear ($1/k^2$)</td>
<td>One gradient</td>
</tr>
<tr>
<td></td>
<td>Quasi-Newton</td>
<td>Superlinear</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Newton method</td>
<td>Sublinear ($1/k$), Quadratic</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strongly convex, smooth $f \in \mathcal{F}_{L,\mu}^2(\mathbb{R}^P)$</td>
<td>Gradient descent</td>
<td>Linear ($e^{-k}$)</td>
<td>One gradient</td>
</tr>
<tr>
<td></td>
<td>Accelerated GD</td>
<td>Linear ($e^{-k}$)</td>
<td>One gradient</td>
</tr>
<tr>
<td></td>
<td>Quasi-Newton</td>
<td>Superlinear</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Newton method</td>
<td>Linear ($e^{-k}$), Quadratic</td>
<td></td>
</tr>
</tbody>
</table>

Accelerated gradient descent:

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

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

for some proper choice of $\alpha$ and $\gamma_{k+1}$. 
Performance of optimization algorithms

A non-exhaustive comparison:

<table>
<thead>
<tr>
<th>Assumptions on $f$</th>
<th>Algorithm</th>
<th>Convergence rate</th>
<th>Iteration complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lipschitz-gradient $f \in \mathcal{F}^{2,1}_L(\mathbb{R}^P)$</td>
<td>Gradient descent</td>
<td>Sublinear $(1/k)$</td>
<td>One gradient</td>
</tr>
<tr>
<td></td>
<td>Accelerated GD</td>
<td>Sublinear $(1/k^2)$</td>
<td>One gradient</td>
</tr>
<tr>
<td></td>
<td>Quasi-Newton</td>
<td>Superlinear</td>
<td>One gradient, rank-2 update</td>
</tr>
<tr>
<td></td>
<td>Newton method</td>
<td>Sublinear $(1/k)$, Quadratic</td>
<td>One gradient, one linear system</td>
</tr>
<tr>
<td>Strongly convex, smooth $f \in \mathcal{F}^{2,1}_{L,\mu}(\mathbb{R}^P)$</td>
<td>Gradient descent</td>
<td>Linear $(e^{-k})$</td>
<td>One gradient</td>
</tr>
<tr>
<td></td>
<td>Accelerated GD</td>
<td>Linear $(e^{-k})$</td>
<td>One gradient</td>
</tr>
<tr>
<td></td>
<td>Quasi-Newton</td>
<td>Superlinear</td>
<td>One gradient, rank-2 update</td>
</tr>
<tr>
<td></td>
<td>Newton method</td>
<td>Linear $(e^{-k})$, Quadratic</td>
<td>One gradient, one linear system</td>
</tr>
</tbody>
</table>

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

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

where $B_k^{-1}$ is updated at each iteration by adding a rank-2 matrix.
Performance of optimization algorithms

A non-exhaustive comparison:

<table>
<thead>
<tr>
<th>Assumptions on $f$</th>
<th>Algorithm</th>
<th>Convergence rate</th>
<th>Iteration complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f \in \mathcal{F}_L^{2,1}(\mathbb{R}^P)$</td>
<td>Gradient descent</td>
<td>Sublinear ($1/k$)</td>
<td>One gradient</td>
</tr>
<tr>
<td></td>
<td>Accelerated GD</td>
<td>Sublinear ($1/k^2$)</td>
<td>One gradient</td>
</tr>
<tr>
<td></td>
<td>Quasi-Newton</td>
<td>Superlinear</td>
<td>One gradient, rank-2 update</td>
</tr>
<tr>
<td></td>
<td>Newton method</td>
<td>Sublinear ($1/k$), Quadratic</td>
<td>One gradient, one linear system</td>
</tr>
<tr>
<td>Strongly convex, smooth $f \in \mathcal{F}_{L,\mu}^{2,1}(\mathbb{R}^P)$</td>
<td>Gradient descent</td>
<td>Linear ($e^{-k}$)</td>
<td>One gradient</td>
</tr>
<tr>
<td></td>
<td>Accelerated GD</td>
<td>Linear ($e^{-k}$)</td>
<td>One gradient</td>
</tr>
<tr>
<td></td>
<td>Quasi-Newton</td>
<td>Superlinear</td>
<td>One gradient, rank-2 update</td>
</tr>
<tr>
<td></td>
<td>Newton method</td>
<td>Linear ($e^{-k}$), Quadratic</td>
<td>One gradient, one linear system</td>
</tr>
</tbody>
</table>

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

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