Introduction to unconstrained optimization - gradient-based methods (cont.)

Jussi Hakanen
Post-doctoral researcher
jussi.hakanen@jyu.fi
People generally remember...
(learning activities)

- 10% of what they read
- 20% of what they hear
- 30% of what they see
- 50% of what they see and hear
- 70% of what they say and write
- 90% of what they do.

People are able to...
(learning outcomes)

- Define
- List
- Describe
- Explain
- Demonstrate
- Apply
- Practice
- Analyze
- Define
- Create
- Evaluate

Cone of learning by Edgar Dale
Model algorithm for unconstrained minimization

1) [Test for convergence.] If conditions are satisfied, stop. The solution is $x^h$.

2) [Compute a search direction.] Compute a non-zero vector $d^h \in \mathbb{R}^n$ which is the search direction.

3) [Compute a step length.] Compute $\alpha_h > 0$, the step length, for which it holds that

   $$f(x^h + \alpha_h d^h) < f(x^h).$$

4) [Update the estimate for minimum.] Set

   $$x^{h+1} = x^h + \alpha_h d^h, \quad h = h + 1$$

   and go to step 1.

Reminder: Important steps

Computing a step length $\alpha_h$
- Requirement of $f(x^h + \alpha_h d^h) < f(x^h)$ is not sufficient
- E.g. consider a univariate function $f(x) = x^2$ and a sequence $(-1)^h \left( \frac{1}{2} + 2^{-h} \right)$
- Thus, the step length must produce a “sufficient decrease”

Computing a search direction $d^h \in \mathbb{R}^n$
- Function $f(x)$ decreases the most locally in the direction $-\nabla f(x)$
- If $d^h$ is almost orthogonal to $\nabla f(x^h)$, then $f$ is almost constant along $d^h$ → this must be avoided

When $f$ is bounded from below, a strictly decreasing sequence $\{x^h\}$ converges
- Can converge to a non-optimal point
- If the above situations are avoided, the level set $L(f(x^0)) = \{x | f(x) \leq f(x^0)\}$ is closed and bounded and $f$ is twice continuously differentiable, then $\lim_{h \to \infty} \|\nabla f(x^h)\| = 0$
Examples of gradient-based methods

- Steepest descent
- Newton’s method
- Quasi-Newton method
- Conjugate gradient method

Today: brief reminder

Today: introduce these
Summary: steepest descent & Newton’s method

- **Steepest descent** ($x^{h+1} = x^h - \alpha_h \nabla f(x^h)$)
  - Descent direction: yes, $\nabla f(x^h)^T d^h = -\nabla f(x^h)^T \nabla f(x^h)$
  - Global convergence: yes
  - Local convergence: zig-zag near optimum
  - Computational bottle neck: $\alpha_h$
  - Memory consumption: $O(n)$

- **Newton’s method** ($x^{h+1} = x^h - H(x^h)^{-1} \nabla f(x^h)$)
  - Descent direction: only if $H(x^h)^{-1}$ positive definite, $\nabla f(x^h)^T d^h = -\nabla f(x^h)^T H(x^h)^{-1} \nabla f(x^h)$
  - Global convergence: no
  - Local convergence: yes, quadratic near optimum
  - Computational bottle neck: $H(x^h)^{-1}$
  - Memory consumption: $O(n^2)$
Notes on Newton’s method

Solve \( f(x) = 0 \) with Newton’s (Newton-Rhapson) method:

\[
x^{h+1} = x^h - \frac{f(x^h)}{f'(x^h)}
\]


Solve \( f'(x) = 0 \) with Newton’s method \((F(x) := f'(x))\)

\[
x^{h+1} = x^h - \frac{F(x^h)}{F'(x^h)} = x^h - \frac{f'(x^h)}{f''(x^h)}
\]

→ Newton’s method in nonlinear optimization is equal to solving \( \nabla f(x) = 0 \)!

Usually, the next iterate is solved from

\[
H(x^h)(x^{h+1} - x^h) = -\nabla f(x^h)
\]

– No need to compute \( H(x^h)^{-1} \) explicitly
Idea of quasi-Newton methods

A drawback of Newton’s method is that $H(x^h)$ may not be positive definite.

Quasi-Newton methods try to retain the good local convergence of Newton’s method, i.e., a local quadratic approximation of $f$ without the second derivatives.

- Cf. solving $f'(x) = 0$, left: Newton’s method, right: the secant method.

From Miettinen: Nonlinear optimization, 2007 (in Finnish)
Quasi-Newton method

- Search direction $d^h = -D^h \nabla f(x^h)$, where a symmetric positive definite matrix $D^h$ is updated iteratively
  - $D^h \rightarrow H^{-1}$ when $x^h \rightarrow x^*$
  - $d^h$ is a descent direction since $D^h$ is positive definite

- The step length $\alpha_h$ is optimized by line search or e.g. $\alpha_h = 1$ can be used
  - $x^{h+1} = x^h - \alpha_h D^h \nabla f(x^h)$
Quasi-Newton algorithm

1) Choose the final tolerance $\epsilon > 0$, a starting point $x^1$ and a symmetric pos. def. $D^1$ (e.g. $D^1 = I$ i.e. an identity matrix). Set $y^1 = x^1$ and $h = j = 1$.

2) If $\| \nabla f(y^j) \| < \epsilon$ stop.

3) Set $d^j = -D^j \nabla f(y^j)$. Let $\alpha_j$ be the solution of problem
   $\min_{\alpha \geq 0} f(y^j + \alpha d^j)$. Set $y^{j+1} = y^j + \alpha_j d^j$. If $j = n$, set
   $y^1 = x^{h+1} = y^{n+1}, h = h + 1, j = 1$ and go to 2).

4) Compute $D^{j+1}$. Set $j = j + 1$ and go to 2).
Updating $D^h$

- Exact Hessian is approximated by using only the objective function and gradient values.

- Taylor series: $\nabla f(x^h + d^h) \approx \nabla f(x^h) + H(x^h)d^h$

- Curvature:
  
  $$(d^h)^T H(x^h) d^h \approx (\nabla f(x^h + d^h) - \nabla f(x^h))^T d^h$$

- Curvature information can be approximated without second derivatives!
DFP & BFGS updates

Let $p^h = x^{h+1} - x^h$ and $q^h = \nabla f(x^{h+1}) - \nabla f(x^h)$

- $H(x^h)^{-1}$ is approximated (DFP update)

$$D^{h+1} = D^h + \frac{p^h(p^h)^T}{(p^h)^T q^h} + \frac{D^hq^h(q^h)^TD^h}{(q^h)^TD^hq^h}$$

- $H(x^h)$ is approximated (BFGS update)

$$D^{h+1} = D^h + \left[1 + \frac{(q^h)^TD^hq^h}{(p^h)^Tq^h}\right] \frac{p^h(p^h)^T}{(p^h)^Tq^h} - \frac{D^hq^h(p^h)^T+p^h(q^h)^TD^h}{(p^h)^Tq^h}$$
Notes

- Usually \( D^1 = I \) is used if no other information is available.
- It can be shown that if \( D^1 \) is pos. def. and \( \alpha_h \) is optimized exactly, then the matrices \( D^2, D^3, \ldots \) are pos. def. in both DFP and BFGS updates.
- Updating \( D^h \) is easier in DFP update but BFGS is not so sensitive to rounding errors.
- A drawback of quasi-Newton methods is that they also use lots of memory: \( O(n^2) \).
Newton’s method: example

Figure 4k. Solution path of a modified Newton algorithm on Rosenbrock’s function. Except for the first iteration, the method follows the base of the valley in an almost “optimal” number of steps, given that piecewise linear segments are used.
Quasi-Newton: example

Figure 4m. Solution path of a BFGS quasi-Newton algorithm on Rosenbrock’s function. Like Newton’s method, the algorithm makes good progress at points remote from the solution. The worst behaviour occurs near the origin, where the curvature is changing most rapidly.
Conjugate gradient methods

- Developed originally for solving systems of linear equations
  - For minimizing a quadratic function without constraints is equivalent to solving $\nabla f(x) = 0$ (system of linear equations if the resulting matrix is pos. def.)
  - Extended to system of nonlinear equations and nonlinear unconstrained optimization
- Are not as efficient as quasi-Newton methods but they require less memory ($O(n)$)
  - Need only to store gradients
  - Good candidates when solving problems with a large number of variables
Conjugate gradient methods (cont.)

- Idea is to improve convergence properties of steepest descent
  - A search direction is a combination of the current search direction and a previous search direction

- Search direction
  - $d^1 = -\nabla f(x^1)$
  - $d^{h+1} = -\nabla f(x^{h+1}) + \beta_h d^h = \frac{1}{\mu} \{ \mu [-\nabla f(x^{h+1})] + (1 - \mu) d^h \}$
  - $\beta_h$ is a parameter which is chosen s.t. the consecutive search directions are conjugate: Let $A$ be a symmetric $n \times x$ matrix. Directions $d^1, \ldots, d^p$ are ($A$-)conjugate if they are linearly independent and if $(d^i)^T A d^j = 0$ for all $i, j = 1, \ldots, p, i \neq j$. 
Conjugate gradient algorithm

1) Choose the final tolerance $\epsilon > 0$ and a starting point $x^1$. Set $y^1 = x^1$, $d^1 = -\nabla f(y^1)$ and $h = j = 1$.

2) If $\|\nabla f(y^j)\| < \epsilon$ stop.

3) Let $\alpha_j$ be the solution of problem $\min_{\alpha \geq 0} f(y^j + \alpha d^j)$. Set $y^{j+1} = y^j + \alpha_j d^j$. If $j = n$, go to 5).

4) Set $d^{j+1} = -\nabla f(y^{j+1}) + \beta_j d^j$, where e.g. $\beta_j = \frac{\|\nabla f(y^{j+1})\|^2}{\|\nabla f(y^j)\|^2}$ (Fletcher and Reeves method). Set $j = j + 1$ and go to 2).

5) Set $y^1 = x^{h+1} = y^{n+1}$ and $d^1 = -\nabla f(y^1)$. Set $j = 1$, $h = h + 1$ and go to 2).
Steepest descent: example

\[ f(x) = (x_1 - 2)^4 + (x_1 - 2x_2)^2 \]
Conjugate gradient: example

\[ f(x) = (x_1 - 2)^4 + (x_1 - 2x_2)^2 \]
Summary

Quasi-Newton methods
- Improve Newton’s method by guaranteeing a descent direction ($D^h$ pos. def.)
- Not quite as good convergence as Newton’s method
- No need to evaluate 2\textsuperscript{nd} derivatives
- Memory consumption $O(n^2)$
- Best suited for small and medium scale problems

Conjugate gradient methods
- Improve the convergence of steepest descent
- Memory consumption $O(n)$
- Best suited for large scale problems