Practical Optimization: Basic Multidimensional Gradient Methods

László Kozma
Lkozma@cis.hut.fi

Helsinki University of Technology
S-88.4221 Postgraduate Seminar on Signal Processing

22. 10. 2008
Contents

• Recap
  • Basic principles
  • Properties of algorithms
  • One-dimensional optimization

• Multidimensional Optimization
  • Overview
  • Steepest descent
  • Newton method
  • Gauss-Newton method

• Homework
Recap

Basic Principles

- Tools
  - Gradient
  - Hessian
  - Taylor series
- Extrema of functions
  - Weak/strong
  - Local/global
  - Necessary and sufficient conditions
- Stationary points
  - Minimum/maximum/saddle
  - Classify them by characterizing the Hessian
- Convex/concave functions
Recap

Properties of Algorithms

- Point-to-point mappings
  - Iterative: $x_k \mapsto x_{k+1}$
  - Descent: $f(x_{k+1}) < f(x_k)$

- Convergence of an algorithm
  - Convergent
  - Convergent to a solution point

- Rate of convergence:

\[
0 \leq \beta \leq \infty, \\
\beta = \lim_{k \to \infty} \frac{|x_{k+1} - \hat{x}|}{|x_k - \hat{x}|^p}
\]
One Dimensional Optimization

- Basic problem: minimize $F = f(x)$ where $x_L \leq x \leq x_U$ knowing that $f(x)$ has single minimum in this range.
- Search methods: repeatedly reduce bracket
  - Dichotomous search
  - Fibonacci search
  - Golden-Section search
- Approximation methods: approximate function with low-order polynomial
Multidimensional Optimization

Overview

- Constrained optimization: usually reduced to unconstrained
- Unconstrained optimization
  - Search methods
    - Perform only function evaluations
    - Explore parameter space in organized manner
    - Very inefficient, used only when gradient info not available
  - Gradient methods
    - First-order (use g)
    - Second-order (use g and H)
Steepest-Descent Method

Minimize $F = f(x)$ for $x \in E^n$

We have from Taylor series:

$$F + \Delta F = f(x + \delta) \approx f(x) + g^T \delta + \frac{1}{2} \delta^T H \delta$$

$$\Delta F \approx g^T \delta$$

$g = [g_1 g_2 \ldots g_n]^T$

$\delta = [\delta_1 \delta_2 \ldots \delta_n]^T$

$$\Delta F \approx \sum_{i=1}^{n} g_i \delta_i = \|g\| \|\delta\| \cos \theta$$

where $\theta$ is the angle between $g$ and $\delta$
Steepest-descent
Steepest-Descent method

- Assuming $f$ continuous around $x$
- Steepest descent direction: $d = -g$
- Change $\delta$ in $x$ given by $\delta = \alpha d$.
- If $\alpha$ small, will decrease value of $f$
- To obtain maximum reduction, solve one-dim. problem:

$$\minimize_{\alpha} F = f(x + \alpha d)$$

- Usually this search does not give minimizer of original $f$
- Therefore we need to perform it iteratively
Steepest-Descent method

Algorithm 5.1 Steepest-descent algorithm

Step 1
Input $x_0$ and initialize the tolerance $\varepsilon$.
Set $k = 0$.

Step 2
Calculate gradient $g_k$ and set $d_k = -g_k$.

Step 3
Find $\alpha_k$, the value of $\alpha$ that minimizes $f(x_k + \alpha d_k)$, using a line search.

Step 4
Set $x_{k+1} = x_k + \alpha_k d_k$ and calculate $f_{k+1} = f(x_{k+1})$.

Step 5
If $\|\alpha_k d_k\| < \varepsilon$, then do:

Output $x^* = x_{k+1}$ and $f(x^*) = f_{k+1}$, and stop.
Otherwise, set $k = k + 1$ and repeat from Step 2.
Orthogonality of directions
Finding $\alpha$

- Line search (see Chapter 4.)
- Analytical solution:

$$f(x_k + \delta_k) \approx f(x_k) + \delta_k^T g_k + \frac{1}{2} \delta_k^T H_k \delta_k$$

$$\delta_k = -\alpha g_k \text{ (steepest-descent direction)}$$

$$\frac{df(x_k - \alpha g_k)}{d\alpha} = 0$$

$$\alpha = \alpha_k \approx \frac{g_k^T g_k}{g_k^T H_k g_k}$$

Approximation accurate if $\delta_k$ small or $f$ quadratic.
Finding $\alpha$

$$\alpha = \alpha_k \approx \frac{g_k^T g_k}{g_k^T H_k g_k}$$

If Hessian not available, approximate $\alpha_k = \hat{\alpha}$ (for ex. value from previous iteration)

$$\hat{f} \approx f_k - \hat{\alpha} g_k^T g_k + \frac{1}{2} \hat{\alpha}^2 g_k^T H_k g_k$$

$$g_k^T H_k g_k \approx \frac{2(\hat{f} - f_k + \hat{\alpha} g_k^T g_k)}{\hat{\alpha}^2}$$

plug it in $\alpha_k$. 
Convergence of Steepest-descent

Provided that:

• $f(x) \in C^2$ has a local minimiser $x^*$
• Hessian is positive definite at $x^*$
• $x_k$ sufficiently close to $x^*$

$$\frac{f(x_{k+1}) - f(x^*)}{f(x_k) - f(x^*)} \leq \left(\frac{1 - r}{1 + r}\right)^2$$

$$r = \frac{\text{smallest eigenvalue of } H_k}{\text{largest eigenvalue of } H_k}$$

• Linear convergence (rate depends on $H_k$)
• Convergence fast if eigenvalues constant (contours circular)
• Consequence: scaling of variables can help
Newton Method

Quadratic approximation using Taylor-series:

\[
f(x + \delta) \approx f(x) + \sum_{j=1}^{n} \frac{\partial f}{\partial x_i} \delta_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 f}{\partial x_i \partial x_j} \delta_i \delta_j
\]

\[
f(x + \delta) \approx f(x) + g^T \delta + \frac{1}{2} \delta^T H \delta
\]

Differentiate with respect to \( \delta_k (k = 1, 2, \ldots, n) \) and set to 0. We obtain \( g = -H \delta \)

The optimum change \( \delta = -H^{-1} g \)
Newton Method

\[ \delta = -H^{-1}g \] Newton direction.

- Solution exists if
  - Hessian is nonsingular
    - Follows from 2nd order sufficiency conditions at \( x^* \) (if minimum exists and we are close to it)
    - Otherwise \( H \) can be forced to become positive definite (implies non-singular)
  - Taylor approximation valid
    - If this holds (quadratic \( f \)), minimum reached in one step
    - Otherwise iterative approach is needed (similarly to Steepest-descent)
    - If \( H \) not positive definite, update may not yield reduction
Algorithm 5.3  Basic Newton algorithm

Step 1
Input $x_0$ and initialize the tolerance $\varepsilon$.
Set $k = 0$.

Step 2
Compute $g_k$ and $H_k$.
If $H_k$ is not positive definite, force it to become positive definite.

Step 3
Compute $H_k^{-1}$ and $d_k = -H_k^{-1}g_k$.

Step 4
Find $\alpha_k$, the value of $\alpha$ that minimizes $f(x_k + \alpha d_k)$, using a line search.

Step 5
Set $x_{k+1} = x_k + \alpha_k d_k$.
Compute $f_{k+1} = f(x_{k+1})$.

Step 6
If $\|\alpha_k d_k\| < \varepsilon$, then do:
Output $x^* = x_{k+1}$ and $f(x^*) = f_{k+1}$, and stop.
Otherwise, set $k = k + 1$ and repeat from Step 2.
Newton method

- Convergence
  - Initially slow, becomes fast close to the solution
  - Complementary to Steepest-descent
  - Order of convergence: 2
  - Main drawback: $H^{-1}$
Modification of the Hessian

How to make $H_k$ positive definite?

1. Goldfeld, Quandt, Trotter’s method

$$\hat{H}_k = \frac{H_k + \beta I_n}{1 + \beta}$$

- If $H_k$ ok, $\beta$ set to small value, $\hat{H}_k \approx H_k$.
- If $H_k$ not ok, $\beta$ set to large value, $\hat{H}_k \approx I_n$, Newton method reduces to Steepest-descent.
Modification of the Hessian

(2) Zwart’s method

\[ \hat{H}_k = U^T H_k U + \epsilon \]

- Where:
  - \( U^T U = I_n \)
  - \( \epsilon \) diagonal \( n \times n \) matrix with elements \( \epsilon_i \)
  - \( U^T H_k U \) diagonal with elements \( \lambda_i \) (the eigenvalues of \( H_k \)).

- Then \( \hat{H}_k \) diagonal with elements \( \lambda_i + \epsilon_i \)

- We can set:
  - \( \epsilon_i = 0 \) if \( \lambda_i > 0 \)
  - \( \epsilon_i = \delta - \lambda_i \) if \( \lambda_i \leq 0 \)

- This way we ignore components due to negative eigenvalues, while preserving convergence properties

- \( U^T H_k U \) formed by solving \( det(H_k - \lambda I_n) \) which is time-consuming.
Modification of the Hessian

(3) Matthews, Davies method

- Practical algorithm based on Gaussian elimination
- Deduce $D = LH_kLT$ ($D$ diagonal, $L$ lower triangular)
- $H_k$ positive definite iff $D$ positive definite (see earlier)
- If $D$ not positive definite, replace each nonpositive element with a positive element, to obtain $\hat{D}$
- Then $\hat{H}_k = L^{-1}\hat{D}(LT)^{-1}$
- The Newton direction: $d_k = -\hat{H}^{-1}g_k = -L^T\hat{D}^{-1}Lg_k$
- The exact algorithm is somewhat involved
Computation of the Hessian

- Second derivatives might be impossible to compute
- They can be approximated with numerical formulas

\[
\frac{\partial f}{\partial x_1} = \lim_{\delta \to 0} \frac{f(x + \delta_1) - f(x)}{\delta} = f'(x) \quad \text{with} \quad \delta_1 = [\delta \ 0 \ 0 \ \cdots \ 0]^T
\]

\[
\frac{\partial^2 f}{\partial x_1 \partial x_2} = \lim_{\delta \to 0} \frac{f'(x + \delta_2) - f'(x)}{\delta} \quad \text{with} \quad \delta_2 = [0 \ \delta \ 0 \ \cdots \ 0]^T
\]
Gauss-Newton Method

- In many problems we want to optimize several functions in the same time
- \( f = [f_1(x) \, f_2(x) \ldots \, f_m(x)]^T \)
- \( f_p(x) \) for \( p = 1, 2, \ldots, m \) independent functions of \( x \)
- We form a new function: \( F = \sum_{p=1}^{m} f_p(x)^2 = f^T f \)
- Minimizing \( F \) in the traditional way is minimizing \( f_p(x) \) in the least-squares sense
- Useful trick the other way around: if function is sum-of-squares, we can "split" it
- We use Newton’s method with some fancy notation
Gauss-Newton Method

\[
J = \begin{bmatrix}
\frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \ldots & \frac{\partial f_1}{\partial x_n} \\
\frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \ldots & \frac{\partial f_2}{\partial x_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \ldots & \frac{\partial f_m}{\partial x_n}
\end{bmatrix}
\]

\[
F = \sum_{p=1}^{m} f_p(x)^2 = f^T f
\]

\[
\frac{\partial F}{\partial x_i} = \sum_{p=1}^{m} 2 f_p(x) \frac{\partial f_p}{\partial x_i}
\]
Gauss-Newton Method

Or in Matrix form:

\[
\begin{bmatrix}
\frac{\partial F}{\partial x_1} \\
\frac{\partial F}{\partial x_2} \\
\vdots \\
\frac{\partial F}{\partial x_n}
\end{bmatrix}
= 2
\begin{bmatrix}
\frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \ldots & \frac{\partial f_m}{\partial x_1} \\
\frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \ldots & \frac{\partial f_m}{\partial x_2} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_n} & \ldots & \frac{\partial f_m}{\partial x_n}
\end{bmatrix}
\begin{bmatrix}
f_1(x) \\
f_2(x) \\
\vdots \\
f_m(x)
\end{bmatrix}
\]

which says in fact:

\[g_F = 2J^Tf\]
Gauss-Newton Method

Similarly for the second derivatives:

\[
\frac{\partial^2 F}{\partial x_i \partial x_j} = 2 \sum_{p=1}^{m} \frac{\partial f_p}{\partial x_i} \frac{\partial f_p}{\partial x_j} + 2 \sum_{p=1}^{m} f_p(x) \frac{\partial^2 f_p}{\partial x_i \partial x_j}
\]

Neglecting the second derivatives:

\[
\frac{\partial^2 F}{\partial x_i \partial x_j} \approx 2 \sum_{p=1}^{m} \frac{\partial f_p}{\partial x_i} \frac{\partial f_p}{\partial x_j}
\]

We obtain: \( H_F \approx 2J^T J \)
Gauss-Newton Method

- Having obtained $g_F$ and $H_F$ we have:

$$x_{k+1} = x_k - \alpha_k (J^T J)^{-1} (J^T f)$$  \hspace{1cm} (1)

- Notes
  - if $f_p(x_k)$ close to linear (near $x^*$), approximation of Hessian is accurate
  - if $f_p(x_k)$ linear, Hessian is exact, we reach solution in one step

- If $H_F$ singular, same solutions as earlier
- Algorithm proceeds similarly as before
Homework

1. What is the role of the Hessian in the convergence rate of the Steepest-descent method?

2. What are some advantages/disadvantages of the Newton method compared to Steepest-descent method?

3. Minimize $f(x) = x_1^2 + 2x_2^2 + 4x_1 + 4x_2$ using steepest-descent method with initial point $x_0 = [00]^T$. (Hint: find a generic term for the iteration points). Show that the algorithm converges to the global minimum.

4. Sketch the optimization steps for $f(x) = -\ln(1 - x_1 - x_2) - \ln(x_1) - \ln(x_2)$ using basic steepest descent and Newton’s method. [Optional: run the optimization, compare convergence rate, accuracy, effect of initial point].

5. Sketch the optimization steps for $f(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 100(x_1 - x_4)^4$