

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

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

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:

$$0 \leq \beta \leq \infty,$$

$$\beta = \lim_{k \rightarrow \infty} \frac{|x_{k+1} - \hat{x}|}{|x_k - \hat{x}|^p}$$

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 \mathbf{H})

Steepest-Descent Method

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

We have from Taylor series:

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

$$\Delta F \approx \mathbf{g}^T \delta$$

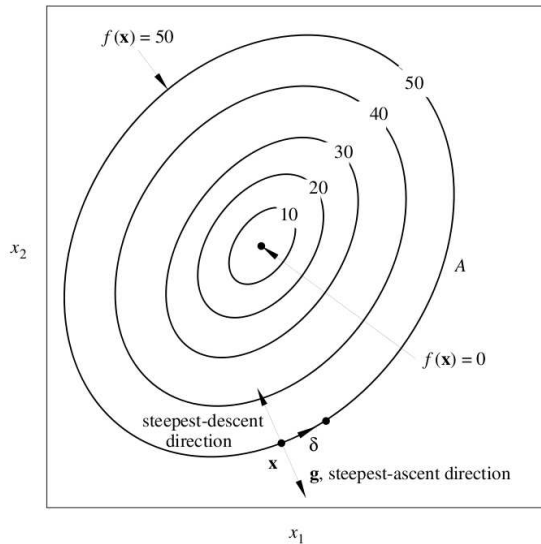
$$\mathbf{g} = [g_1 g_2 \dots g_n]^T$$

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

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

where θ is the angle between \mathbf{g} and δ

Steepest-descent



Steepest-Descent method

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

$$\text{minimize}_{\alpha} F = f(\mathbf{x} + \alpha\mathbf{d})$$

- 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 \mathbf{x}_0 and initialize the tolerance ε .

Set $k = 0$.

Step 2

Calculate gradient \mathbf{g}_k and set $\mathbf{d}_k = -\mathbf{g}_k$.

Step 3

Find α_k , the value of α that minimizes $f(\mathbf{x}_k + \alpha\mathbf{d}_k)$, using a line search.

Step 4

Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{d}_k$ and calculate $f_{k+1} = f(\mathbf{x}_{k+1})$.

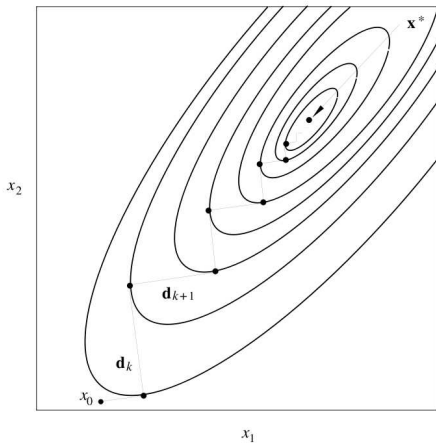
Step 5

If $\|\alpha_k\mathbf{d}_k\| < \varepsilon$, then do:

Output $\mathbf{x}^* = \mathbf{x}_{k+1}$ and $f(\mathbf{x}^*) = f_{k+1}$, and stop.

Otherwise, set $k = k + 1$ and repeat from Step 2.

Orthogonality of directions



Finding α

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

$$f(\mathbf{x}_k + \delta_k) \approx f(\mathbf{x}_k) + \delta_k^T \mathbf{g}_k + \frac{1}{2} \delta_k^T \mathbf{H}_k \delta_k$$

$$\delta_k = -\alpha \mathbf{g}_k \text{ (steepest-descent direction)}$$

$$\frac{df(\mathbf{x}_k - \alpha \mathbf{g}_k)}{d\alpha} = 0$$

$$\alpha = \alpha_k \approx \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T \mathbf{H}_k \mathbf{g}_k}$$

Approximation accurate if δ_k small or f quadratic.

Finding α

$$\alpha = \alpha_k \approx \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T \mathbf{H}_k \mathbf{g}_k}$$

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

$$\hat{f} \approx f_k - \hat{\alpha} \mathbf{g}_k^T \mathbf{g}_k + \frac{1}{2} \hat{\alpha}^2 \mathbf{g}_k^T \mathbf{H}_k \mathbf{g}_k$$

$$\mathbf{g}_k^T \mathbf{H}_k \mathbf{g}_k \approx \frac{2(\hat{f} - f_k + \hat{\alpha} \mathbf{g}_k^T \mathbf{g}_k)}{\hat{\alpha}^2}$$

plug it in α_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^*

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

$$r = \frac{\text{smallest eigenvalue of } \mathbf{H}_k}{\text{largest eigenvalue of } \mathbf{H}_k}$$

- Linear convergence (rate depends on \mathbf{H}_k)
- Convergence fast if eigenvalues constant (contours circular)
- Consequence: scaling of variables can help

Newton Method

Quadratic approximation using Taylor-series:

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

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

Differentiate with respect to $\delta_k (k = 1, 2, \dots, n)$ 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

Newton method

Algorithm 5.3 Basic Newton algorithm

Step 1

Input \mathbf{x}_0 and initialize the tolerance ε .

Set $k = 0$.

Step 2

Compute \mathbf{g}_k and \mathbf{H}_k .

If \mathbf{H}_k is not positive definite, force it to become positive definite.

Step 3

Compute \mathbf{H}_k^{-1} and $\mathbf{d}_k = -\mathbf{H}_k^{-1} \mathbf{g}_k$.

Step 4

Find α_k , the value of α that minimizes $f(\mathbf{x}_k + \alpha \mathbf{d}_k)$, using a line search.

Step 5

Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$.

Compute $f_{k+1} = f(\mathbf{x}_{k+1})$.

Step 6

If $\|\alpha_k \mathbf{d}_k\| < \varepsilon$, then do:

Output $\mathbf{x}^* = \mathbf{x}_{k+1}$ and $f(\mathbf{x}^*) = f_{k+1}$, 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

$$\hat{\mathbf{H}}_k = \frac{\mathbf{H}_k + \beta \mathbf{I}_n}{1 + \beta}$$

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

Modification of the Hessian

(2) Zwart's method

$$\hat{\mathbf{H}}_k = \mathbf{U}^T \mathbf{H}_k \mathbf{U} + \epsilon$$

- Where:
 - $\mathbf{U}^T \mathbf{U} = \mathbf{I}_n$
 - ϵ diagonal $n \times n$ matrix with elements ϵ_i
 - $\mathbf{U}^T \mathbf{H}_k \mathbf{U}$ diagonal with elements λ_i (the eigenvalues of \mathbf{H}_k).
- Then $\hat{\mathbf{H}}_k$ 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
- $\mathbf{U}^T \mathbf{H}_k \mathbf{U}$ formed by solving $\det(\mathbf{H}_k - \lambda \mathbf{I}_n)$ which is time-consuming.

Modification of the Hessian

(3) Matthews, Davies method

- Practical algorithm based on Gaussian elimination
- Deduce $D = LH_kL^T$ (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 $\hat{H}_k = L^{-1}\hat{D}(L^T)^{-1}$
- 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

$$\frac{\partial f}{\partial x_1} = \lim_{\delta \rightarrow 0} \frac{f(\mathbf{x} + \boldsymbol{\delta}_1) - f(\mathbf{x})}{\delta} = f'(\mathbf{x}) \quad \text{with } \boldsymbol{\delta}_1 = [\delta \ 0 \ 0 \ \dots \ 0]^T$$
$$\frac{\partial^2 f}{\partial x_1 \partial x_2} = \lim_{\delta \rightarrow 0} \frac{f'(\mathbf{x} + \boldsymbol{\delta}_2) - f'(\mathbf{x})}{\delta} \quad \text{with } \boldsymbol{\delta}_2 = [0 \ \delta \ 0 \ \dots \ 0]^T$$

Gauss-Newton Method

- In many problems we want to optimize several functions in the same time
- $\mathbf{f} = [f_1(\mathbf{x}) f_2(\mathbf{x}) \dots f_m(\mathbf{x})]^T$
- $f_p(\mathbf{x})$ for $p = 1, 2, \dots, m$ independent functions of x
- We form a new function: $F = \sum_{p=1}^m f_p(\mathbf{x})^2 = \mathbf{f}^T \mathbf{f}$
- Minimizing F in the traditional way is minimizing $f_p(\mathbf{x})$ 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

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

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

$$\frac{\partial F}{\partial x_i} = \sum_{p=1}^m 2f_p(x) \frac{\partial f_p}{\partial x_i}$$

Gauss-Newton Method

Or in Matrix form:

$$\begin{bmatrix} \frac{\partial F}{\partial x_1} \\ \frac{\partial F}{\partial x_2} \\ \vdots \\ \frac{\partial F}{\partial x_n} \end{bmatrix} = 2 \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_n} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix}$$

which says in fact:

$$\mathbf{g}_F = 2\mathbf{J}^T \mathbf{f}$$

Gauss-Newton Method

Similarly for the second derivatives:

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

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: $\mathbf{H}_F \approx 2\mathbf{J}^T \mathbf{J}$

Gauss-Newton Method

- Having obtained \mathbf{g}_F and \mathbf{H}_F we have:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k (\mathbf{J}^T \mathbf{J})^{-1} (\mathbf{J}^T \mathbf{f}) \quad (1)$$

- Notes

- if $f_p(\mathbf{x}_k)$ close to linear (near \mathbf{x}^*), approximation of Hessian is accurate
- if $f_p(\mathbf{x}_k)$ linear, Hessian is exact, we reach solution in one step
- If \mathbf{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 $f(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 100(x_1 - x_4)^4$

Reference

Andreas Antonio and Wu-Sheng Lu, Practical Optimization Algorithms and Engineering Applications, Springer, 2007