Computational statistics

Optimizing smooth multivariate functions

Thierry Denœux

February 2017

Contents of the course (Part I)

1. Optimizing smooth univariate functions: Bisection, Newton's method, Fisher scoring, secant method
2. Optimizing smooth multivariate functions: nonlinear Gauss-Seidel iteration, gradient methods, Newton's method, Fisher scoring, Gauss-Newton method, ascent algorithms, discrete Newton method, quasi-Newton methods
3. Combinatorial optimization: local search, ascent algorithms, simulated annealing, genetic algorithms
4. Expectation-Maximization (EM) algorithm for maximizing the likelihood or posterior density

Multivariate optimization for smooth g

- Let $g : \boldsymbol{x} \in \mathbb{R}^p \to \mathbb{R}$
- Can use analogous stopping criteria:

$$D(\mathbf{x}^{(t+1)}, \mathbf{x}^{(t)}) < \epsilon, \quad \frac{D(\mathbf{x}^{(t+1)}, \mathbf{x}^{(t)})}{D(\mathbf{x}^{(t)}, \mathbf{0})} < \epsilon,$$

  or

$$\frac{D(\mathbf{x}^{(t+1)}, \mathbf{x}^{(t)})}{D(\mathbf{x}^{(t)}, \mathbf{0}) + \epsilon} < \epsilon.$$

  for $D(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{p} |u_i - v_i|$ or $D(\mathbf{u}, \mathbf{v}) = \sqrt{\sum_{i=1}^{p} (u_i - v_i)^2}$.
- Same strategy of iterative approximation. We will extend previous methods and introduce new options.

Overview

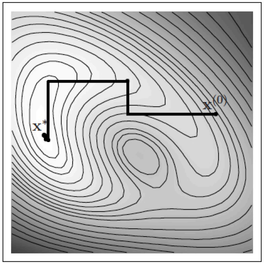
Cyclic coordinate ascent

- Also called backfitting or Gauss-Seidel iteration. One key application is for fitting additive models, GAMs, etc.
- Idea: transform a p-dimensional optimization problem into p univariate optimization problems. How ?

Cyclic coordinate ascent

- Also called backfitting or Gauss-Seidel iteration. One key application is for fitting additive models, GAMs, etc.
- Idea: transform a p-dimensional optimization problem into p univariate optimization problems. How ?
- Approach: optimize g with respect to each component of $\mathbf{x}$ successively, fixing all other components to their last obtained value.

Algorithm



Case $p = 2$:

- Initialize $x_1 = x_1^{(0)}$
- Find $x_2^{(1)} = \arg\max_{x_2} g(x_1^{(0)}, x_2)$
- Find $x_1^{(1)} = \arg\max_{x_1} g(x_1, x_2^{(1)})$
- Find $x_2^{(2)} = \arg\max_{x_2} g(x_1^{(1)}, x_2)$
- $\vdots$

Cyclic coordinate ascent: pros and cons

- Advantages:
  1. Simplifies a potentially difficult problem
  2. Solution of each univariate problem is easier and more stable
- Drawbacks
  1. Convergence is not guaranteed
  2. Can be slow
- For hard problems (high dimension, complex function shape), we need more sophisticated optimization procedures.

Overview

Gradient ascent

- Gradient methods are based on the gradient

$$\mathbf{g}'(\mathbf{x}) = \left(\frac{\partial g(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial g(\mathbf{x})}{\partial x_p}\right)^T,$$

  which indicates the direction of steepest ascent of function g at $\mathbf{x}$.
- The steepest ascent method uses the update equation

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \alpha^{(t)}\mathbf{g}'(\mathbf{x}^{(t)}),$$

  where $\alpha^{(t)}$ is the step size at iteration t.
- How to determine the step size?

Ascent property

- For small enough $\alpha^{(t)}$, we have $g(\mathbf{x}^{(t+1)}) > g(\mathbf{x}^{(t)})$.
- Proof: we have

$$\begin{aligned} g(\mathbf{x}^{(t+1)}) - g(\mathbf{x}^{(t)}) &= g(\mathbf{x}^{(t)} + \alpha^{(t)}\mathbf{g}'(\mathbf{x}^{(t)})) - g(\mathbf{x}^{(t)}) \\ &= \alpha^{(t)}\mathbf{g}'(\mathbf{x}^{(t)})^T\mathbf{g}'(\mathbf{x}^{(t)}) + o(\alpha^{(t)}), \end{aligned}$$

where the second equality follows from the linear Taylor expansion

$$g(\mathbf{x}^{(t)} + \alpha^{(t)}\mathbf{g}'(\mathbf{x}^{(t)})) = g(\mathbf{x}^{(t)}) + \alpha^{(t)}\mathbf{g}'(\mathbf{x}^{(t)})^T\mathbf{g}'(\mathbf{x}^{(t)}) + o(\alpha^{(t)}).$$

- Therefore, ascent can be ensured by choosing $\alpha^{(t)}$ sufficiently small, yielding

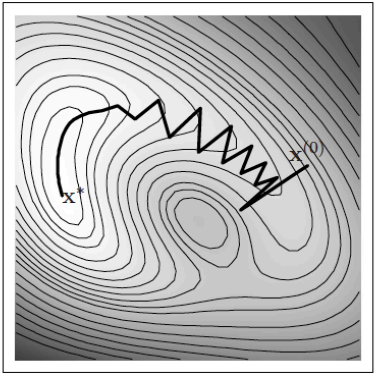
$$g(\mathbf{x}^{(t+1)}) - g(\mathbf{x}^{(t)}) > 0$$

from (??) since $o(\alpha^{(t)})/\alpha^{(t)} \to 0$ as $\alpha^{(t)} \to 0$.

Determining $\alpha^{(t)}$

- Choosing $\alpha^{(t)}$ very small guarantees ascent, but can result in very slow convergence.
- We need a strategy to adapt $\alpha^{(t)}$, making it as large as possible, while ensuring uphill steps.
- Backtracking: attempt a step for, say, $\alpha^{(t)} = 1$;
  - If it is downhill, backtrack and reduce (e.g.,halve) $\alpha^{(t)}$.
  - If the step is still downhill, continue halving $\alpha^{(t)}$ until a sufficiently small step is found to be uphill.
- Step adaptation: attempt a step with the current value $\alpha^{(t)}$;
  - If it is downhill, backtrack and set $\alpha^{(t+1)} = b\alpha^{(t)}$ with $b < 1$.
  - If it is uphill, keep the last move and set $\alpha^{(t+1)} = a\alpha^{(t)}$ with $a > 1$

Example



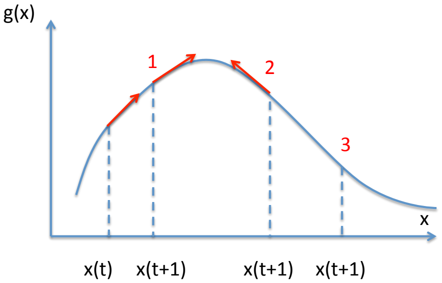
Steepest ascent with backtracking, using $\alpha = 0.25$ initially at each step
The steepest ascent direction is not necessarily the wisest, and backtracking doesn't prevent oversteps

Silva-Almeida algorithm

- Update rule:

$$x_j^{(t+1)} = x_j^{(t)} - \alpha_j^{(t)} \frac{\partial g(\mathbf{x}^{(t)})}{\partial \mathbf{x}_j}$$

- A learning rate $\alpha_j^{(t)}$ is adapted separately for each weight x_j.



1. Case 1: accept the change and set $\alpha_j^{(t+1)} = a\,\alpha_j^{(t)}$ with $a > 1$;
2. Case 2: accept the change and set $\alpha_j^{(t+1)} = b\,\alpha_j^{(t)}$ with $b > 1$;
3. Case 3: backtrack and $\alpha_j^{(t+1)} = c\,\alpha_j^{(t)}$ with $c < 1$ for all j.

- Typically, $a = 1.2$, $b = 0.8$, $c = 0.5$.

Overview

Multivariate Newton's method

- In the Silva-Almeida method, the step at each step is no longer in the direction of the gradient.
- Indeed, the gradient direction is not always the best. For instance, if g is quadratic,

$$g(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$$

  with A negative definite, the unique maximum can be found in one step from any starting point $\mathbf{x}^{(0)}$ by

$$x^* = -\mathbf{A}^{-1}\mathbf{b} = \mathbf{x}^{(0)} - \mathbf{g}''(\mathbf{x}^{(0)})^{-1}\mathbf{g}'(\mathbf{x}^{(0)}) \tag{1}$$

  where $\mathbf{g}''(\mathbf{x}) = \left(\frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j}\right)$ is the $p \times p$ Hessian matrix of g at $\mathbf{x}$
- Newton's method: at each time step, approximate $g(\mathbf{x})$ around $\mathbf{x}^{(t)}$ by a second-order Taylor series expansion, and use update equation (1).

Multivariate Newton's method and Fisher scoring

- 2nd order approximation of $g(\mathbf{x})$ around $\mathbf{x}^{(t)}$:

$$g(\mathbf{x}) \approx g(\mathbf{x}^{(t)}) + (\mathbf{x}-\mathbf{x}^{(t)})^T \mathbf{g}'(\mathbf{x}^{(t)}) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^{(t)})^T \mathbf{g}''(\mathbf{x}^{(t)})(\mathbf{x}-\mathbf{x}^{(t)}).$$

- Setting $\mathbf{g}'(\mathbf{x}) = \mathbf{0}$, we get the update equation

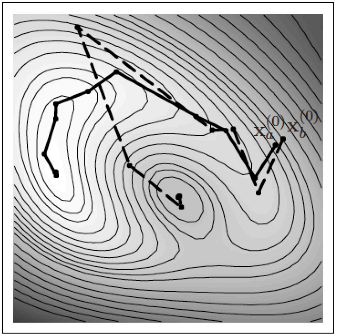
$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \mathbf{g}''(\mathbf{x}^{(t)})^{-1}\mathbf{g}'(\mathbf{x}^{(t)}).$$

- Fisher scoring:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \mathbf{I}(\boldsymbol{\theta}^{(t)})^{-1}\ell'(\boldsymbol{\theta}^{(t)}),$$

where $\mathbf{I}(\boldsymbol{\theta}) = -\mathbb{E}\{\ell''(\boldsymbol{\theta})\}$ is the Fisher information matrix at $\boldsymbol{\theta}$.

Example



Two runs starting from $\mathbf{x}_a^{(0)}$ and $\mathbf{x}_b^{(0)}$ are shown. These converge to the true maximum and to a local minimum, respectively.
Newton's method is not guaranteed to walk uphill. It is not guaranteed to find a local maximum. Step length matters even when step direction is good.

Newton-like methods

- Some very effective methods rely on updating equations of the form

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - (\mathbf{M}^{(t)})^{-1}\mathbf{g}'(\mathbf{x}^{(t)})$$

  where $\mathbf{M}^{(t)}$ is a $p \times p$ matrix approximating the Hessian, $\mathbf{g}''(\mathbf{x}^{(t)})$.
- Two issues:
  - We want to avoid calculating Hessian if it is computationally expensive or analytically difficult
  - We want to guarantee uphill steps

Ascent algorithms

- If we use the updating increment

$$\mathbf{h}^{(t)} = -\alpha^{(t)} \left[\mathbf{M}^{(t)}\right]^{-1} \mathbf{g}'(\mathbf{x}^{(t)}).$$

  then any positive definite matrix $-\mathbf{M}^{(t)}$ will ensure ascent for a sufficiently small $\alpha^{(t)}$
- Backtracking can be used, as in the steepest ascient method.
- Steepest ascent is recovered as a special case, with $\mathbf{M}^{(t)} = -\mathbf{I}$.
- Fisher scoring is another special case with $-\mathbf{M}^{(t)} = \mathbf{I}(\boldsymbol{\theta}^{(t)})$. Since $\mathbf{I}(\boldsymbol{\theta}^{(t)})$ is positive semi-definite, backtracking with Fisher scoring avoids stepping downhill.

Discrete Newton method

- To avoid calculating the Hessian, one could resort to an analogue of the 1-dimensional secant method.
- For example,

$$\mathbf{M}_{ij}^{(t)} = \frac{g_i'(\mathbf{x}^{(t)} + h_{ij}^{(t)}\mathbf{e}_j) - g_i'(\mathbf{x}^{(t)})}{h_{ij}^{(t)}}$$

  where $g_i'(\mathbf{x}) = dg(\mathbf{x})/dx_i$ is the ith element of $\mathbf{g}'(\mathbf{x})$, $\mathbf{e}_j$ is the p-vector with a 1 in the jth position and zeros elsewhere, and $h_{ij}^{(t)}$ are some constants.
- $h_{ij}^{(t)} = h$ for all (i, j) and t leads to linear convergence order: $\beta = 1$.
- Alternatively, $h_{ij}^{(t)} = x_j^{(t)} - x_j^{(t-1)}$ for all i gives superlinear convergence.

Quasi-Newton methods

- The discrete Newton method strategy is computationally burdensome because $\mathbf{M}^{(t)}$ is wholly updated at every step.
- A more efficient approach can be designed based on the direction of the most recent step. From a first order Taylor series expansion of $\mathbf{g}'$ at $\mathbf{x}^{(t)}$, we have

$$\mathbf{g}'(\mathbf{x}^{(t+1)}) - \mathbf{g}'(\mathbf{x}^{(t)}) \approx \mathbf{g}''(\mathbf{x}^{(t)})(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)})$$

- $\mathbf{M}^{(t+1)}$ satisfies the secant condition if

$$\mathbf{g}'(\mathbf{x}^{(t+1)}) - \mathbf{g}'(\mathbf{x}^{(t)}) = \mathbf{M}^{(t+1)}(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}). \tag{2}$$

- Goal: generate $\mathbf{M}^{(t+1)}$ from $\mathbf{M}^{(t)}$ in a manner that requires few calculations and satisfies (2), while learning about the curvature of $\mathbf{g}'$ in the direction of the most recent step.

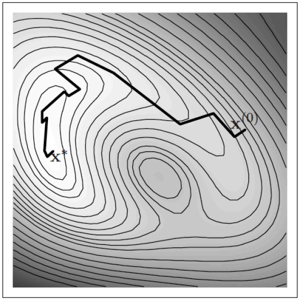
BFGS method

- The widely used BFGS method updates matrix $\mathbf{M}^{(t+1)}$ so as to satisfy the secant condition. It is defined by the following update equation

$$\mathbf{M}^{(t+1)} = \mathbf{M}^{(t)} - \frac{\mathbf{M}^{(t)}\mathbf{z}^{(t)}(\mathbf{M}^{(t)}\mathbf{z}^{(t)})^T}{(\mathbf{z}^{(t)})^T\mathbf{M}^{(t)}\mathbf{z}^{(t)}} + \frac{\mathbf{y}^{(t)}(\mathbf{y}^{(t)})^T}{(\mathbf{z}^{(t)})^T\mathbf{y}^{(t)}}$$

  where $\mathbf{z}^{(t)} = \mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}$ and $\mathbf{y}^{(t)} = \mathbf{g}'(\mathbf{x}^{(t+1)}) - \mathbf{g}'(\mathbf{x}^{(t)})$.
- The BFGS update confers hereditary positive definiteness: if $-\mathbf{M}^{(t)}$ is positive definite, so is $-\mathbf{M}^{(t+1)}$.
- Backtracking is normally used.

Example



Quasi-Newton optimization with the BFGS update and backtracking to ensure ascent.

Convergence of quasi-Newton methods is generally superlinear, but not quadratic. These are powerful and popular methods, available, for example, in the R function `optim()`.

Overview

Gauss-Newton method

Basic idea

- For nonlinear least squares problems with observed data $(y_i, \mathbf{z}_i)$ for $i = 1, \ldots, n$ and model

$$Y_i = f(\mathbf{z}_i, \boldsymbol{\theta}) + \epsilon_i$$

  for some non-linear function, f, and random error, ϵ_i.

- We seek to estimate $\boldsymbol{\theta}$ by maximizing an objective function

$$g(\boldsymbol{\theta}) = -\sum_{i=1}^{n} (y_i - f(\mathbf{z}_i, \boldsymbol{\theta}))^2 .$$

- Newton's method approximates g via Taylor series. The Gauss-Newton approach approximates f itself by its linear Taylor series expansion about $\boldsymbol{\theta}^{(t)}$.

Gauss-Newton method

Linearized reformulation

- We have

$$f(\mathbf{z}_i, \boldsymbol{\theta}) \approx f(\mathbf{z}_i, \boldsymbol{\theta}^{(t)}) + (\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)})^T \mathbf{f}'(\mathbf{z}_i, \boldsymbol{\theta}^{(t)})$$

  where for each i, $\mathbf{f}'(\mathbf{z}_i, \boldsymbol{\theta}^{(t)})$ is the column vector of partial derivatives of f with respect to $\theta_j^{(t)}$, for $j = 1, \ldots, p$, evaluated at $(\mathbf{z}_i, \boldsymbol{\theta}^{(t)})$.
- Now, instead of $g(\boldsymbol{\theta})$, we maximize

$$\begin{aligned}\tilde{g}(\boldsymbol{\theta}) &= -\sum_{i=1}^{n} \left(y_i - f(\mathbf{z}_i, \boldsymbol{\theta}^{(t)}) - (\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)})^T \mathbf{f}'(\mathbf{z}_i, \boldsymbol{\theta}^{(t)}) \right)^2 \\ &= -\sum_{i=1}^{n} \left(x_i^{(t)} - (\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)})^T \mathbf{a}_i^{(t)} \right)^2\end{aligned}$$

  with respect to $\boldsymbol{\theta}$, with $x_i^{(t)} = y_i - f(\mathbf{z}_i, \boldsymbol{\theta}^{(t)})$, and define $\mathbf{a}_i^{(t)} = \mathbf{f}'(\mathbf{z}_i, \boldsymbol{\theta}^{(t)})$.

Gauss-Newton method

Update equation

- Then the approximated problem can be re-expressed as minimizing the squared residuals of the linear regression model

$$\mathbf{X}^{(t)} = \mathbf{A}^{(t)}(\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)}) + \boldsymbol{\epsilon}$$

  where $\mathbf{X}^{(t)}$ and $\boldsymbol{\epsilon}$ are column vectors whose *i*th elements consist of $X_i^{(t)}$ and ϵ_i, respectively. Similarly, $\mathbf{A}^{(t)}$ is a matrix whose *i*th row is $(\mathbf{a}_i^{(t)})^T$.
- This is a linear regression problem! Thus,

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \left((\mathbf{A}^{(t)})^T\mathbf{A}^{(t)}\right)^{-1}(\mathbf{A}^{(t)})^T\mathbf{x}^{(t)}.$$

- Requires no computation of Hessian.
- Works best when the model fits fairly well and f is not severely nonlinear.

Overview

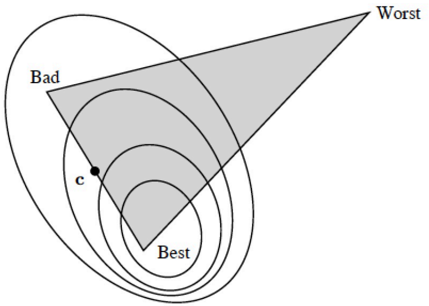
Nelder-Mead algorithm

Main idea

- An algorithm that does not require the calculation of $g(\mathbf{x})$ or $g''(\mathbf{x})$.
- Idea: evaluation g at $p+1$ points $\mathbf{x}_1, \ldots, \mathbf{x}_p$ forming a simplex.
- This simplex defines a region, which is iteratively reshaped by replacing the worst point (vertex) by a better one.

Nelder-Mead algorithm

Definitions



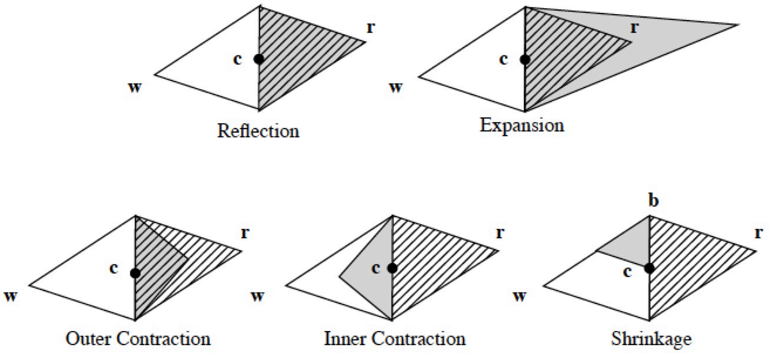
Let

- $\mathbf{x}_{best}$: vertex with highest value of g
- $\mathbf{x}_{worst}$: vertex with lowest value of g
- $\mathbf{x}_{bad}$: 2nd worst vertex
- Best face: face opposite to $\mathbf{x}_{worst}$, $\mathbf{c}$ its centroid.

Nelder-Mead algorithm

Transformations of a vertex

Five possible transformations of a vertex:



Nelder-Mead algorithm

Basic algorithm

- The location of the new vertex (replacing $\mathbf{x}_{worst}$) is based on the reflection vertex $\mathbf{x}_r = \mathbf{c} + \alpha_r(\mathbf{c} - \mathbf{x}_{worst})$, usually $\alpha_r = 1$
- If $g(\mathbf{x}_{bad}) < g(\mathbf{x}_r) < g(\mathbf{x}_{best})$: keep $\mathbf{x}_r$ as the new vertex
- If $g(\mathbf{x}_r) > g(\mathbf{x}_{best})$: try an expansion step
- If $g(\mathbf{x}_r) < g(\mathbf{x}_{bad})$: try a contraction step

Nelder-Mead algorithm

Expansion, contraction, shrinking

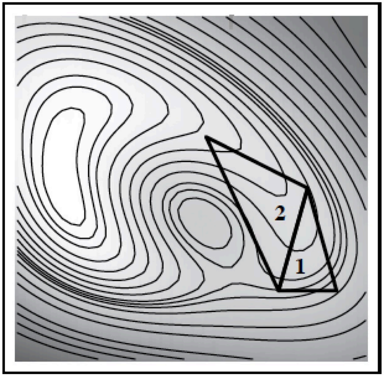
- Expansion: let $\mathbf{x}_e = \mathbf{c} + \alpha_e(\mathbf{x}_r - \mathbf{c})$, usually $\alpha_e = 2$
  - If $g(\mathbf{x}_e) > g(\mathbf{x}_r)$: set $\mathbf{x}_e$ as the new vertex
  - Otherwise, keep $\mathbf{x}_r$
- Contraction:
  - If $g(\mathbf{x}_r) > g(\mathbf{x}_{worst})$: outer contraction. Let $\mathbf{x}_o = \mathbf{c} + \alpha_c(\mathbf{x}_r - \mathbf{c})$, usually $\alpha_c = 0.5$.
    - If $g(\mathbf{x}_o) > g(\mathbf{x}_r)$: keep $\mathbf{x}_o$
    - Otherwise: perform a shrink transformation
  - If $g(\mathbf{x}_r) \leq g(\mathbf{x}_{worst})$: inner contraction. Let $\mathbf{x}_i = \mathbf{c} + \alpha_c(\mathbf{x}_{worst} - \mathbf{c})$.
    - If $g(\mathbf{x}_i) > g(\mathbf{x}_{worst})$: keep $\mathbf{x}_i$
    - Otherwise: perform a shrink transformation
- Shrink transformation: all vertices except $\mathbf{x}_{best}$ are shrunk toward $\mathbf{x}_{best}$:

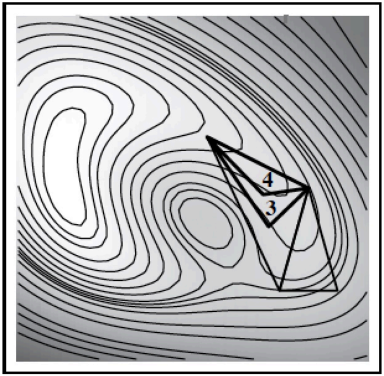
$$\mathbf{x}'_j = \mathbf{x}_{best} + \alpha_s(\mathbf{x}_j - \mathbf{x}_{best},$$

usually $\alpha_s = 0.5$.

Nelder-Mead algorithm

Example





Nelder-Mead algorithm

Example