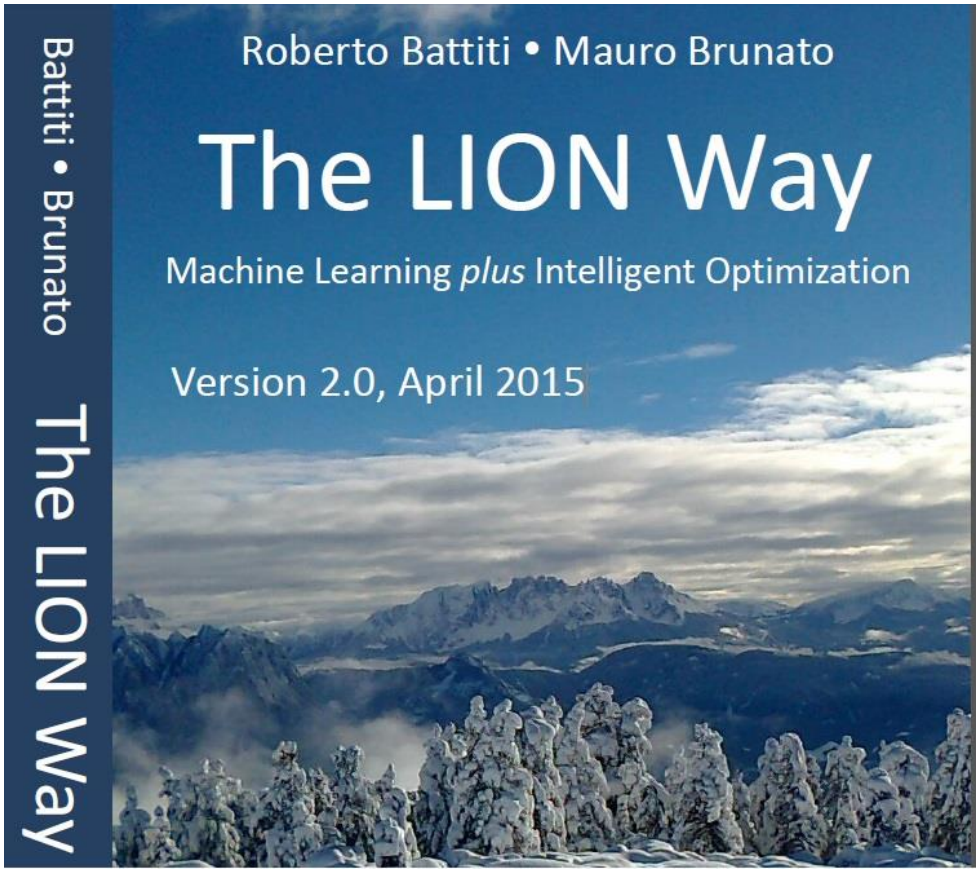
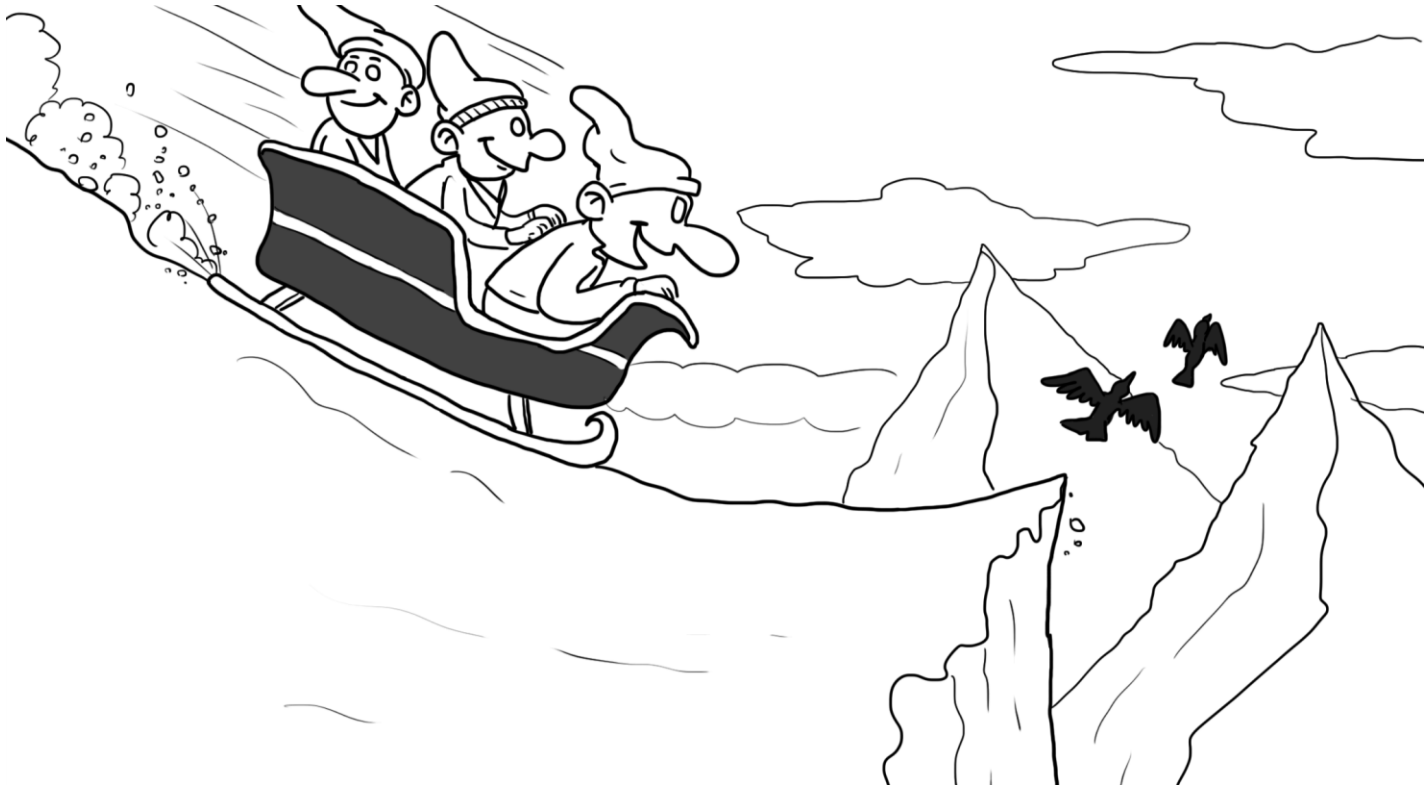


ROBERTO BATTITI, MAURO BRUNATO.
The LION Way: Machine Learning plus *Intelligent Optimization*.
LIONlab, University of Trento, Italy,
Apr 2015

http://intelligent-optimization.org/LIONbook

Automated improvements by local steps

In this world - I am gonna walk
Until my feet - refuse to take me any longer
Yes I' m gonna walk - and walk some more.
(Macy Gray and Zucchero Fornaciari)



Optimization

- Most problems can be cast as finding the optimal value for a suitable **objective function**, subject to constraints
- Methods to **optimize** functions are the source of power for most problem solving and decision making activities
- Maximizing = identifying the input values causing the maximum output value

Two related problems: Minimization and root finding

Nonlinear equations problem: solving a set of nonlinear equations

$$\text{Given } F : \mathbb{R}^n \longrightarrow \mathbb{R}^n$$

$$\text{find } x^* \in \mathbb{R}^n \text{ such that } F(x^*) = 0 \in \mathbb{R}^n$$

Unconstrained minimization

$$\text{Given } f : \mathbb{R}^n \longrightarrow \mathbb{R}$$

$$\text{find } x^* \in \mathbb{R}^n \text{ such that } f(x^*) \leq f(x) \text{ for every } x \in \mathbb{R}^n.$$

Optimization and learning

- Optimization for learning:

Select, among a class of models, one that is **most consistent** with the data provided, e.g., minimizing the sum of squared differences

- Learning for optimization

Learning is used in optimization algorithms to **build local models** of the function to be optimized

Derivative-based techniques for optimization in one dimension

- Root finding: How does one find a point where a *differentiable* function f(x) is equal to zero?

Start with a point **sufficiently close** to the target and iterate the following:

1. Find a **local solvable model**
2. **Solve** the local model

Newton's method

- Let f(x) be a differentiable function. The local model around a point x_c can be derived from **Taylor series approximation**

$$f(x) = f(x_c) + f'(x_c)(x - x_c) + \frac{f''(x_c)(x - x_c)^2}{2!} + \dots$$

- A local model around the current estimate x_c is therefore

$$M_c(x) = f(x_c) + f'(x_c)(x - x_c)$$

Root finding: Newton's method

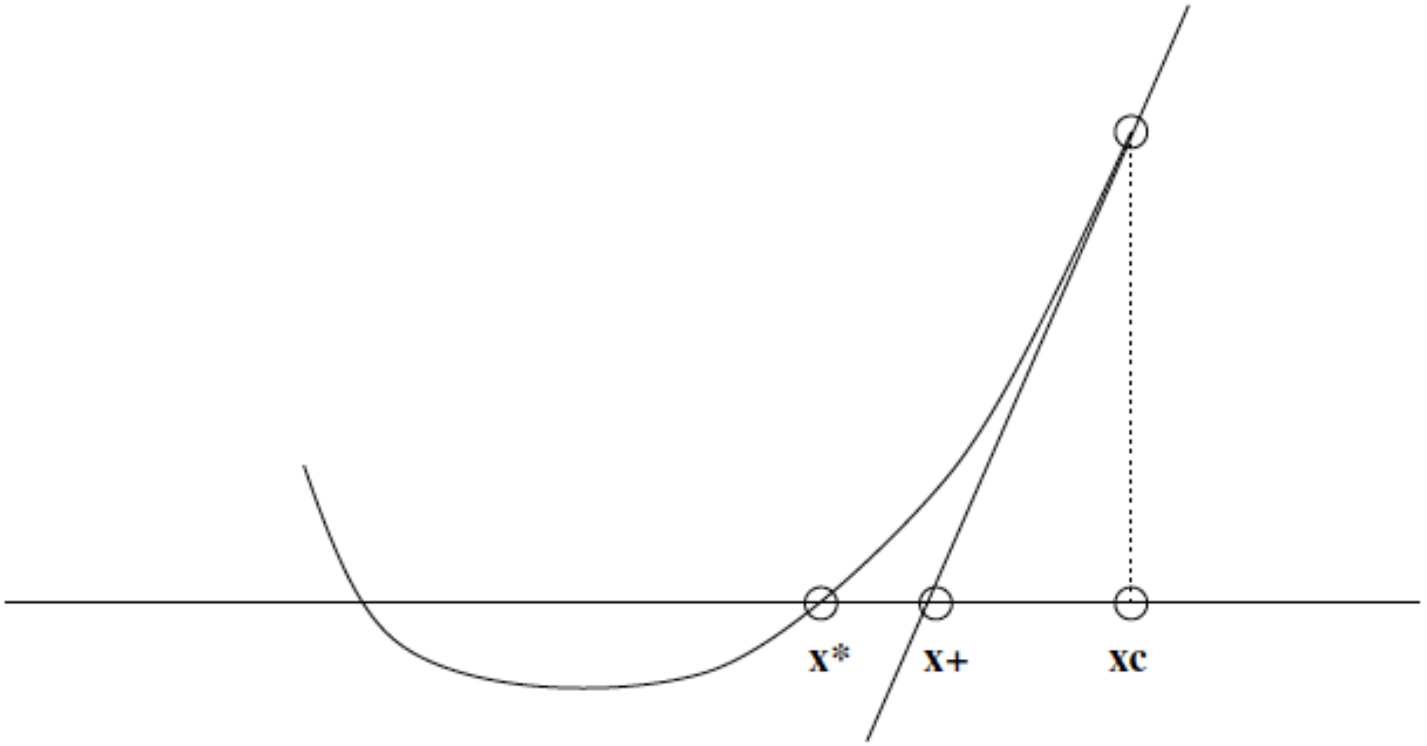


Figure 18.1: Local model for Newton's method.

Newton's method

- by finding the **root of the model** one gets a prescription for the next value x_+ of the current estimate

$$x_+ = x_c - \frac{f(x_c)}{f'(x_c)}$$

- Iterating the two steps, under some hypothesis, x_c converges to the solution

Newton's method: convergence

Definition 1 (Lipschitz continuity) *A function g is Lipschitz continuous with constant γ in a set X ($g \in Lip_\gamma(X)$) if for every $x, y \in X$:*

$$|g(x) - g(y)| \leq \gamma |x - y|.$$

The following lemma easily follows from the previous definition

Lemma 1 *Let $f' \in Lip_\gamma(D)$ for an open interval D. Then for any $x, y \in X$:*

$$|f(y) - f(x) - f'(x)(y - x)| \leq \gamma \frac{(x - y)^2}{2}.$$

Newton's method: convergence

Using lemma one, it is easy to proof the following

Theorem 1 *Let $f : D \longrightarrow \mathbb{R}$ for open interval D, $f' \in Lip_\gamma(D)$ (Lipschitz), $|f'(x)| \geq \rho$ (derivative bounded away from zero) in D.*

If $f(x) = 0$ has a solution $x^* \in D$, then the solution can be found by Newton method if the starting point x_0 is sufficiently close:
there is $\eta > 0$ such that if $|x_0 - x^*| < \eta$, the sequence:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

exists and converges to x^*. In addition:

$$|x_{k+1} - x^*| \leq \frac{\gamma}{2\rho}|x_k - x^*|^2.$$

Newton's method: convergence

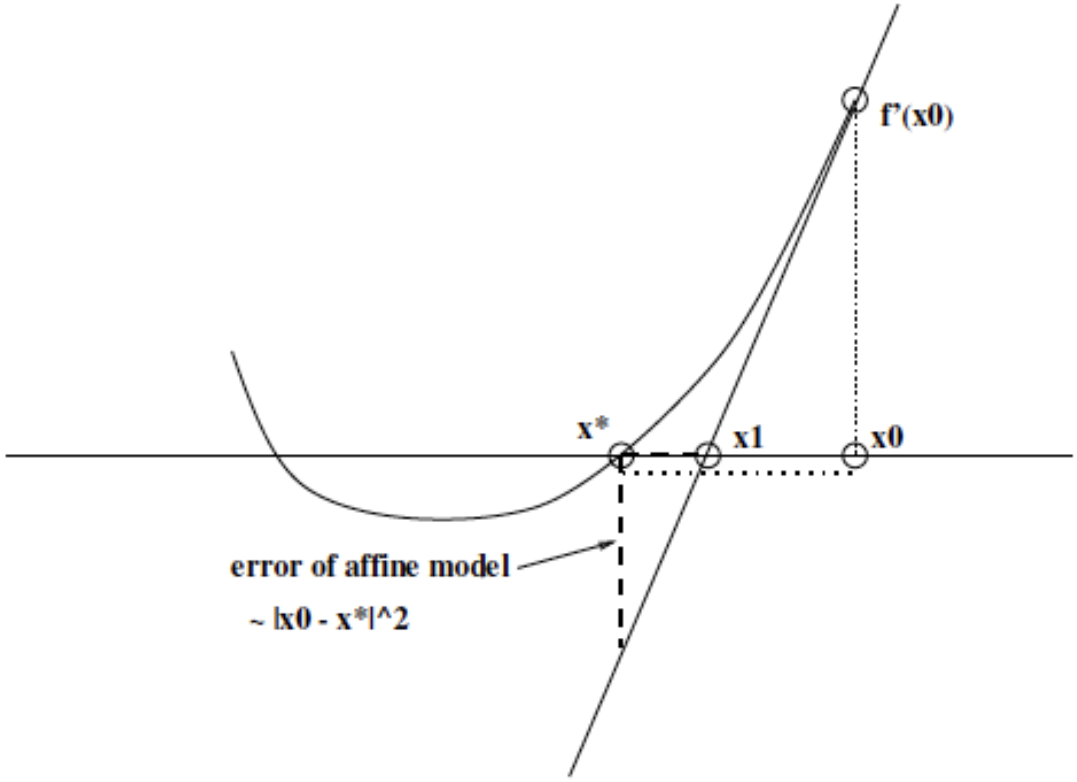


Figure 18.2: Convergence is guaranteed if the starting point x_0 is close to x^*.

Root finding: Bisection method

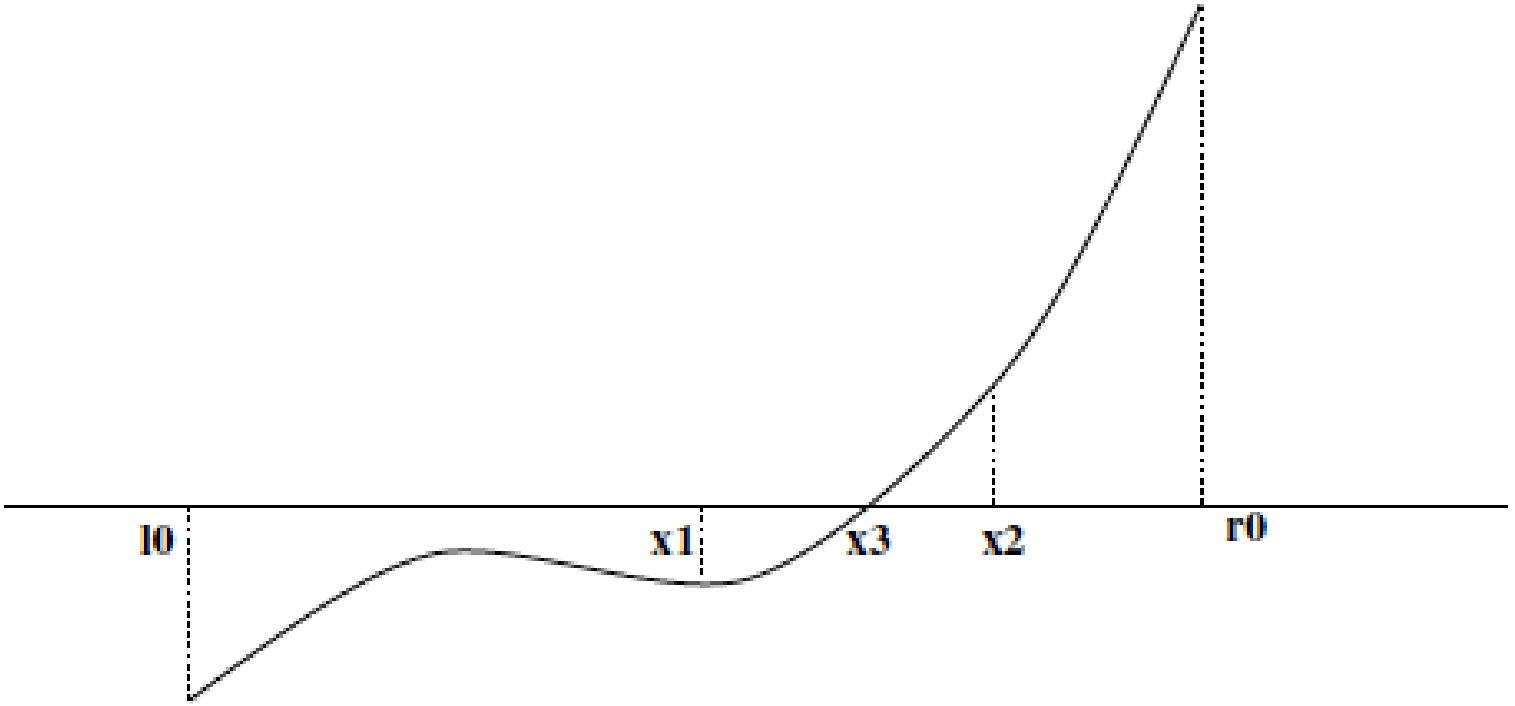


Figure 18.3: The bisection method.

Root finding: Bisection method (2)

What if no guarantee of starting sufficiently close? Bisection method is more robust!

1. subdivide an initial interval into two parts

2. observe the value at the **middle point**

3. continue the search by considering only the left or the right sub-interval

Newton and bisection: pros & cons

1. Newton:
   - quadratic convergence
   - only **locally** convergent
2. Bisection
   - simple and effective
   - **globally** convergent
   - logarithmic convergence
   - cannot be extended to higher dimensions

Hybrid methods combine global convergence and fast local convergence

Hybrid methods

- Generic scheme: combine global convergence and fast local convergence

```
1. function hybrid_quasi_newton (f : ℝ → ℝ, x_0)
2.     while not finished
3.         Make local model of f around x_k, find x_N that solves the model;
4.         if x_{k+1} is acceptable then move
5.         else pick x_{k+1} by using a safe global strategy.
```

Figure 18.5: The hybrid quasi-Newton algorithm.

Backtracking

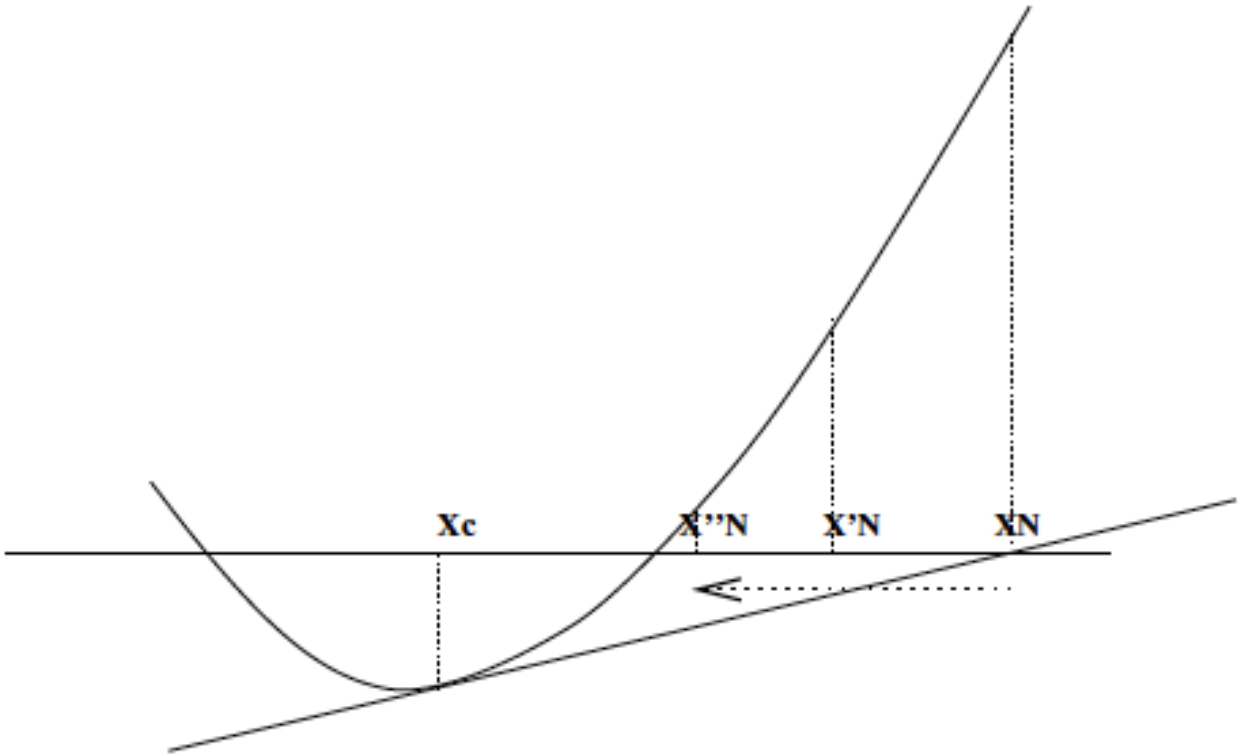


Figure 18.4: Backtracking: Newton step gives the direction

if Newton's step leads too far, beyond the position of the root, one reverts the direction coming back closer to the root position

Approximate derivative with secant

- If derivatives are not available one can approximate them with the secant:

$$a_c = \frac{f(x_c) - f(x_-)}{x_c - x_-}.$$

- A convergence theorem is valid:

Theorem 2 *Let $f : D \longrightarrow \mathbb{R}$ for open interval D, $f' \in Lip_\gamma(D)$ (Lipschitz), $|f'(x)| \geq \rho$ (derivative bounded away from zero) in D.*

If $f(x) = 0$ has a solution $x^* \in D$, then there exist positive constants η, η' such that if $0 < |h_k| \leq \eta'$ and if $|x_0 - x^*| < \eta$, then the sequence

$$x_{k+1} = x_k - \frac{f(x_k)}{a_k}\ ,\ a_k = \frac{f(x_k + h_k) - f(x_k)}{h_k}$$

converges $q-linearly$ to x^*.

Minimization of differentiable functions

- If a differentiable function f attains a minimum at x^* , then $f'(x^*)=0$.
- The problem can be reduced to **finding a root of the derivative** function (necessary condition, but not sufficient)
- We know how to do it! (just apply Newton, or bisection, or a hybrid algorithm, to f')

Solving models in more dimensions

- Solving the **local quadratic model** in higher dimension amounts to solving a quadratic form.
- Newton's method now requires that the gradient of the model be equal to zero.
- Given a step s the quadratic model is

$$Q(s) = \sum_{i=1}^{n} g_i s_i + \sum_{i=1}^{n}\sum_{j=1}^{n} H_{ij} s_i s_j \equiv g^T s + \frac{1}{2} s^T H s$$

Positive-definite quadratic forms

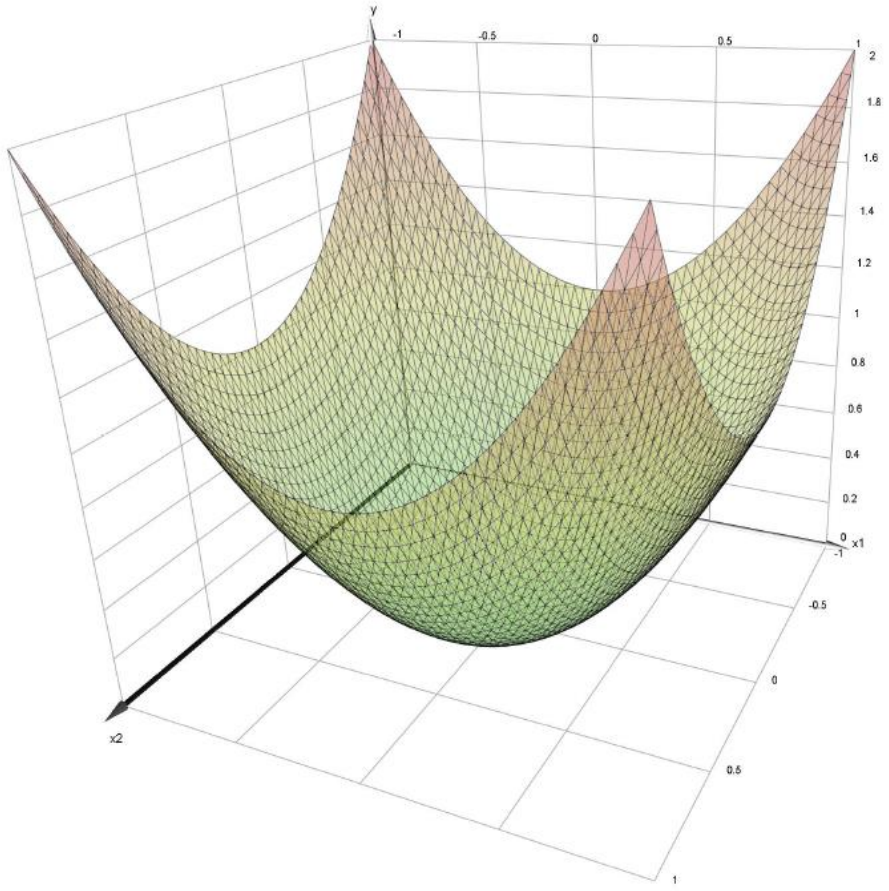


Figure 18.6: Quadratic positive definite f of two variables.

Solving models in more dimensions(2)

- After deriving the **gradient**, one demands

$$\begin{aligned} \nabla Q(s) &= 0 = g + Hs; \\ Hs^N &= -g \quad \text{(Newton equation)} \end{aligned}$$

- The solution of the linear system can be found in one step of cost $O(n^3)$ for the standard matrix inversion

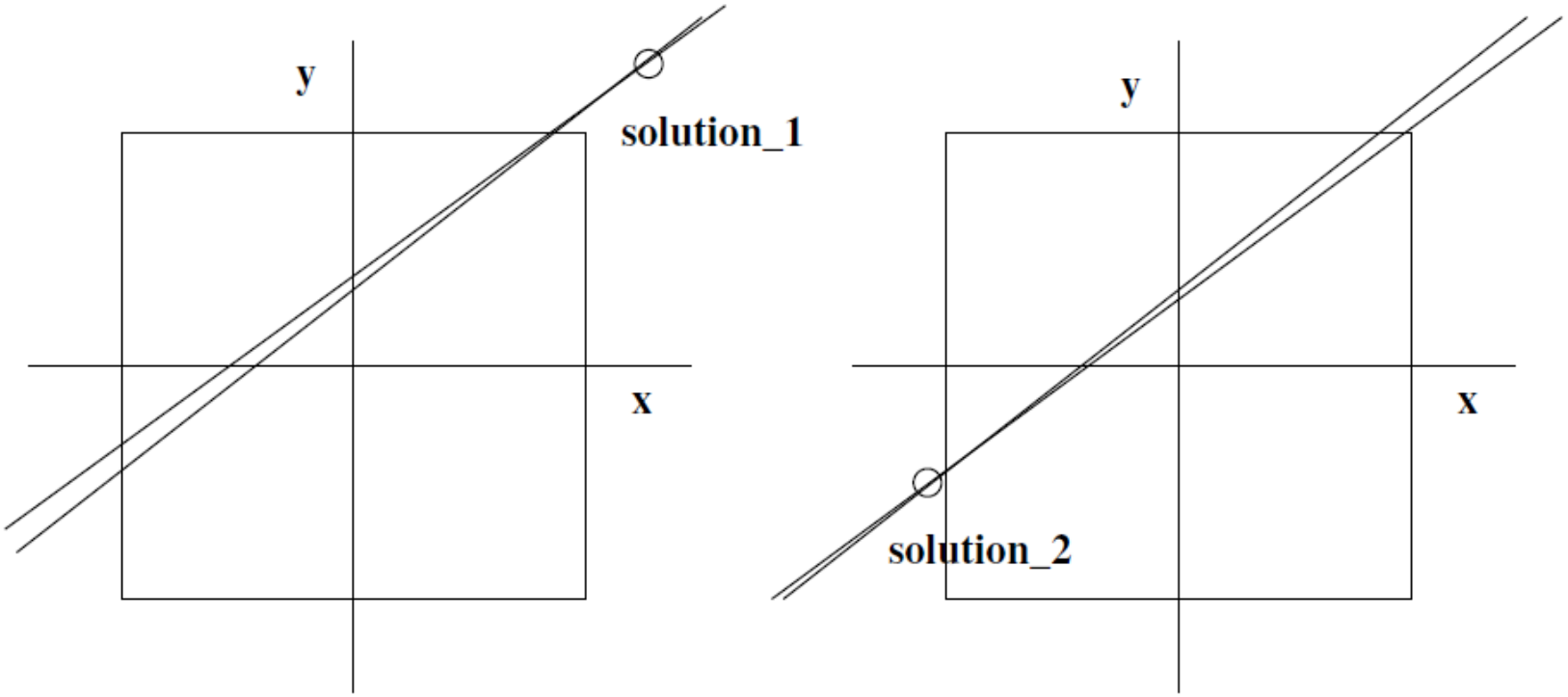
Numerical instability

- computation carried out by computers has to deal with issues of numerical stability

- **Errors can accumulate** in a dangerous way, leading to wrong numerical solutions

- The solution can be very **sensitive to small changes** in the data (**ill conditioning**)

Ill conditioning



Ill-conditioning: solution is very **sensitive to changes in the data**. In this case two linear equations are very similar and a small change in the line direction is sufficient to shift the solution by a large amount.

Quantifying ill-conditioning

- The condition number k(H) of a matrix H is defined as ||H|| $||H^{-1}||$

$$\|H\| = \max_x(\|Hx\|/\|x\|)$$

- k(H) measures the sensitivity of the solution of a linear system to finite-precision arithmetic

Quantify ill conditioning(2)

- If a linear system H x = b is **perturbed** with an error proportional to ε

$$(H + \epsilon F)s(\epsilon) = g + \epsilon f$$

- the relative error in the solution can be bounded as:

$$\frac{\|s(\epsilon) - s\|}{\|s\|} \leq \kappa(H) \left(\frac{\|\epsilon F\|}{\|H\|} + \frac{\|\epsilon f\|}{\|g\|}\right) + O(\epsilon^2)$$

Cholesky factorization

- For symmetric and positive definite matrices, Cholesky factorization is an extremely **stable** way to find a triangular decomposition.

$$H = LDL^T$$

- With L lower triangular, D diagonal with strictly positive elements.
- Since the diagonal is strictly positive, we can write

$$H = LD^{1/2}D^{1/2}L^T = \bar{L}\bar{L}^T = R^T R$$

- where R is a general upper triangular matrix.

Cholesky decomposition: construction

- R can be computed directly from the element-by-element equality:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} r_{11} & & & \\ r_{21} & r_{22} & & \\ \vdots & \vdots & \ddots & \\ r_{n1} & r_{n2} & \dots & r_{nn} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ & r_{22} & \dots & r_{2n} \\ & & \ddots & \vdots \\ & & & r_{nn} \end{pmatrix}$$

- This process requires 1/6 n^3 multiplications and additions and n square roots

Solving a linear system with Cholesky factorization

- Once the Cholesky factorization is available, the original equation becomes

$$R^T R s = g$$

- It can be solved by back-substitution

$$\begin{aligned} R^T s_1 &= -g \quad \text{use forward substitution;} \\ Rs &= s_1 \quad \text{use backward substitution.} \end{aligned}$$

- The cost for solving the equation is $O(n^2)$: the dominant cost is in the factorization

Gradient or steepest descent

Two gradient-descent experts on the mountains surrounding Trento, Italy.

Gradient descent

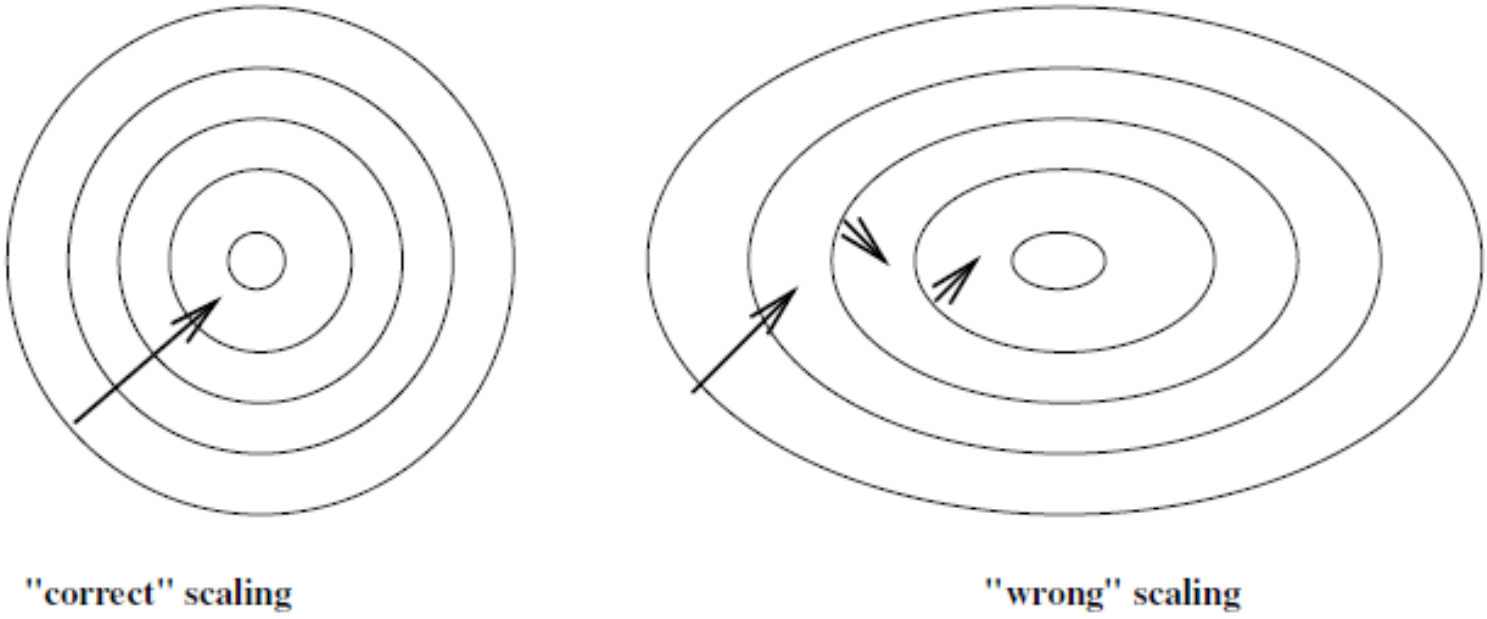
- finding the minimum of the quadratic model by matrix inversion is often neither efficient nor robust
- **steepest descent** is a possible strategy to gradually improve a starting solution
- moving along the negative gradient, the function decreases **for sufficiently small values of the step**

$$x_+ = x_c - \epsilon \nabla f \qquad\qquad f(x_+) < f(x_c)$$

Gradient descent: pros & cons

- simple to implement
- intuitive interpretations (think about a drop of water on a surface, or about a skier)
- used in many applications
- ε has to be carefully chosen
- **no global vision** is available to guide the search, only local information.
- If the matrix is ill-conditioned , the gradient direction does *not* point towards the optimal value

Gradient not always the “best” direction



The gradient is not always an appropriate direction: the trajectory can **zig-zag (right figure)**.

Conjugate gradient

- Conjugate gradient method aims at prescribing a set of directions along which one should iteratively optimize the function
- Two directions are **mutually conjugate** with respect to the matrix H if

$$p_i^T H p_j = 0 \quad \text{when} \quad i \neq j.$$

- After minimizing in direction p_i , the gradient at the minimizer will be perpendicular to p_i
- The second minimization is in direction p_{i+1} : the change of the gradient along this direction is $g_{i+1} - g_i = \alpha H p_{i+1}$ and it is perpendicular to p_i
- being the gradient perpendicular to p_i, **the previous minimization is not spoiled**

Conjugate gradient: construction of the directions

- Define $y_k = g_{k+1} - g_k$
- The first search direction p_1 is given by the negative gradient $-g_1$. The sequence x_k of approximations to the minimizer is defined by:

$$x_{k+1} = x_k + \alpha_k p_k,$$
$$p_{k+1} = -g_{k+1} + \beta_k p_k,$$

- g_k is the gradient, α_k is chosen to minimize E along the p_k and β_k is given by:

$$\beta_k = \frac{y_k^T g_{k+1}}{g_k^T g_k} \qquad \text{(Polak-Ribiere choice)},$$

or by:

$$\beta_k = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} \qquad \text{(Fletcher-Reeves choice)}.$$

Nonlinear optimization in more dimensions

- Newton's method in more dimensions consists of solving the quadratic model

$$m_c(x_c + p) = f(x_c) + \nabla f(x_c)^T p + \frac{1}{2} p^T \nabla^2 f(x_c) p,$$

function `multi_dimensional_newton` ($f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{x}_0 \in \mathbb{R}^n$)
f is twice continuously differentiable
while not finished
solve $\nabla^2 f(\boldsymbol{x}_c) s^N = -\nabla f(\boldsymbol{x}_c)$;
$\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + s^N$.

Figure 18.10: Newton method in more dimensions.

Newton's method in higher dimensions: possible problems

- Conditions for convergence:
  - initial point is close to the minimizer x,
  - Hessian is positive definite at the minimizer

Problems if the Hessian is not positive definite, singular or ill-conditioned

Modified Newton's methods change the local model to obtain a sufficiently positive-definite and non-singular matrix.

Combine a fast tactical local method with a robust strategic method to assure global convergence

Global convergence through line searches

- Global convergence is obtained by adopting line searches along the identified direction
- if H is positive definite, Newton's direction is a descent direction

$$\frac{df}{d\lambda}(x_c + \lambda s^N) = \nabla f(x_c)^T s^T = -\nabla f(x_c)^T H_c^{-1} \nabla f(x_c) < 0$$

- How do we ensure global convergence?

  f value must decrease by a sufficient amount w.r.t the step length
  step must be long enough
  search direction must remain not orthogonal to the gradient

Global convergence through line searches(2)

- In order to guarantee the above points we can resort to Armijo and Goldstein conditions

1.
$$f(x_c + \lambda_c p) \leq f(x_c) + \alpha \lambda_c \nabla f(x_c)^T p,$$
where $\alpha \in (0, 1)$ and $\lambda_c > 0$;

2.
$$\nabla f(x_c + \lambda_c p)^T p \geq \beta \nabla f(x_c)^T p,$$
where $\beta \in (\alpha, 1)$.

Global convergence through line searches(3)

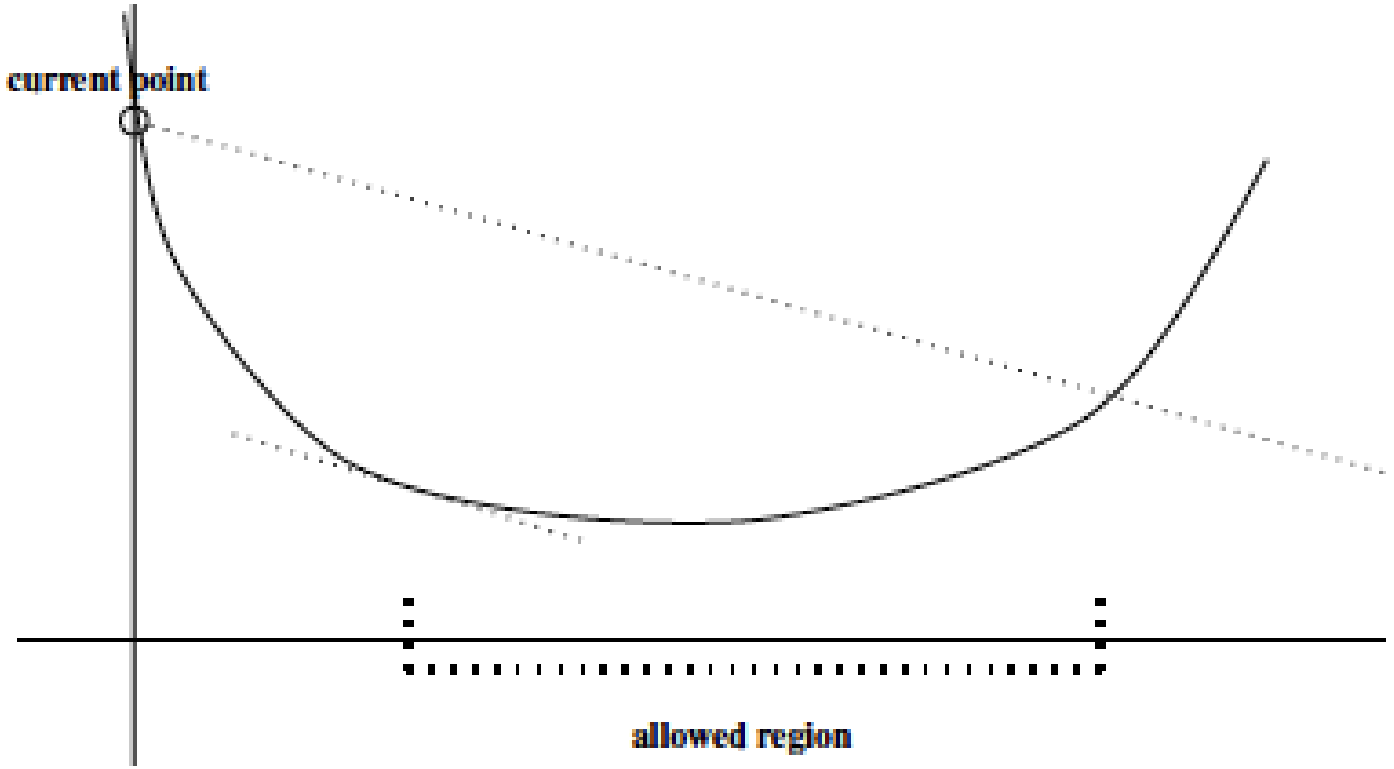
- If the Armijo-Goldstein conditions are satisfied at each iteration and if the error is bounded below, one has the following global convergence property:

$$\lim_{k\to\infty} \nabla f(x_c) = 0,$$

- provided that each step is away from orthogonality to the gradient:

$$\lim_{k\to\infty} \nabla f(x_c) s_k / \|s_k\| \neq 0.$$

- If the Armijo-Goldstein conditions are maintained, one-dimensional searches lead to global convergence



Visualization of Armijo - Goldstein conditions.

Cure for indefinite Hessians

- If the Hessian is indefinite one can use the modified Cholesky method
- It consists in **adding to H a simple diagonal matrix**:

$$H' = \nabla^2 f(x_c) + \mu_c I, \;\; \mu_c > 0$$

and performing a Cholesky decomposition on the modified Hessian

- This amounts to adding a positive definite quadratic form to our original model.

Relations with model-trust region methods

- In **model-trust region** methods the model is trusted only within a region, that is updated by using the experience accumulated during the search process.

Theorem 3 *Suppose that we are looking for the step s_c that solves:*

$$\min m_c(x_c + s) \quad = \quad f(x_c) + \nabla f(x_c)^T s + \frac{1}{2} s^T H_c s$$

$$\textit{subject to } \|s\| \leq \delta_c.$$

The above problem is solved by:

$$s(\mu) = -(H_c + \mu I)^{-1} \nabla f(x_c), \tag{18.14}$$

for the unique $\mu \geq 0$ such that the step has the maximum allowed length ($\|s(\mu)\| = \delta_c$), unless the step with $\mu = 0$ is inside the trusted region ($\|s(0)\| \leq \delta_c$), in which case $s(0)$, the Newton step, is the solution.

Relations with model-trust region methods(2)

- The diagonal modification of the Hessian is a compromise between gradient descent and Newton's method :

μ tends to zero ⟶ the step tends to coincide with Newton's step,

μ is large ⟶ the step tends to be proportional to the negative gradient:

Secant methods in higher dimension

- Secant techniques are useful if the Hessian is not available or costly to calculate.
- Let the current and next point be x_c and x_+, respectively, and let's define $s_c = x_+ - x_c$ and

$$y_c = \nabla f(x_+) - \nabla f(x_c)$$

- The analogous "secant equation" is

$$H_+ s_c = y_c.$$

Secant methods in higher dimension

- The above equation does not determine a unique H_+ but leaves the freedom to choose from a $(n^2 - n)$ dimensional affine subspace
- The equation will not be used to determine but to **update** a previously available approximation
- One can find the matrix in $Q(s_c, y_c)$ that is closest to the previously available matrix

Secant methods in higher dimension: Broyden's update

- The resulting Broyden's update is

$$(H_+)_1 = H_c + \frac{(y_c - H_c s_c) s_c^T}{s_c^T s_c}.$$

- Warning: it may be not symmetric, but
- Iterating Broyden projection and a projection onto the subspace of symmetric matrixes one obtains a sequence of matrixes converging to a solution that is both in $Q(s_c, y_c)$ and symmetric!

Secant methods in higher dimension: Powell's update

- The symmetric secant update of Powell is given by a composition of Broyden's update and a projection onto the subspace of the symmetric matrixes

$$H_+ = H_c + \frac{(y_c - H_c s_c)s_c^T + s_c(y_c - H_c s_c)^T}{s_c^T s_c} - \frac{< y_c - H_c s_c, s_c > s_c s_c^T}{(s_c^T s_c)^2}.$$

- For the update to be also positive definite we can resort to the Broyden, Fletcher, Goldfarb, and Shanno (BFGS) update, that reads

$$H_+ = H_c + \frac{y_c y_c^T}{y_c^T s_c} - \frac{H_c s_c s_c^T H_c}{s_c^T H_c S_c}.$$

Second-order methods with linear complexity

- Complexity:

-Computing the exact Hessian: $O(n^2)$ operations, $O(n^2)$ memory
-Determining the search direction: $O(n^3)$ operations

computation and memory requirements to find the search direction can be reduced to O(n): calculate some second-order information by starting from the last gradients.

One-step method

- The one-step method requires only vectors computed from gradients. The new search direction p_+ is obtained as:

$$p_+ = -g_c + A_c s_c + B_c y_c,$$

- Where

$$A_c = -\left(1 + \frac{y_c^T y_c}{s_c^T y_c}\right)\frac{s_c^T g_c}{s_c^T y_c} + \frac{y_c^T g_c}{s_c^T y_c} \;;\; B_c = \frac{s_c^T g_c}{s_c^T y_c}.$$

s_c, g_c and y_c are respectively last step, gradient and difference of gradients.

One-step method

- The one-step method requires only vectors computed from gradients. The new search direction p_+ is obtained as:

$$p_+ = -g_c + A_c s_c + B_c y_c,$$

- Where

$$A_c = -\left(1 + \frac{y_c^T y_c}{s_c^T y_c}\right)\frac{s_c^T g_c}{s_c^T y_c} + \frac{y_c^T g_c}{s_c^T y_c} \;;\; B_c = \frac{s_c^T g_c}{s_c^T y_c}.$$

s_c, g_c and y_c are respectively last step, gradient and difference of gradients.

Derivative-free techniques: the **Reactive Affine Shaker** (RAS)

- Partial derivative may not be computable in some cases (the function may not be differentiable, or the computation may be too hard)

- In this case, we use optimization methods based only on the knowledge of **function values**

Adaptive random search: general scheme

Choose an initial point in the configuration space and an initial search region surrounding it and repeat:

1. Generate a **new candidate point** sampling the search region according to a given probability measure
2. If the value of the function at the new point is is greater then the current (failure to improve), **compress** the search region, otherwise **expand** it
3. If the sample is successful the **new point** becomes the current point, and the search region is moved so that the current point is at its center

RAS: adaptation of the sampling region

- Reactive Affine Shaker (RAS): self-adaptive and derivative-free optimization method
- Main design criterion: **adaptation of a search region by an affine transformation**
- The modification takes into account the **local knowledge derived from trial points** generated with a uniform probability in the search region.

RAS algorithm pseudo-code

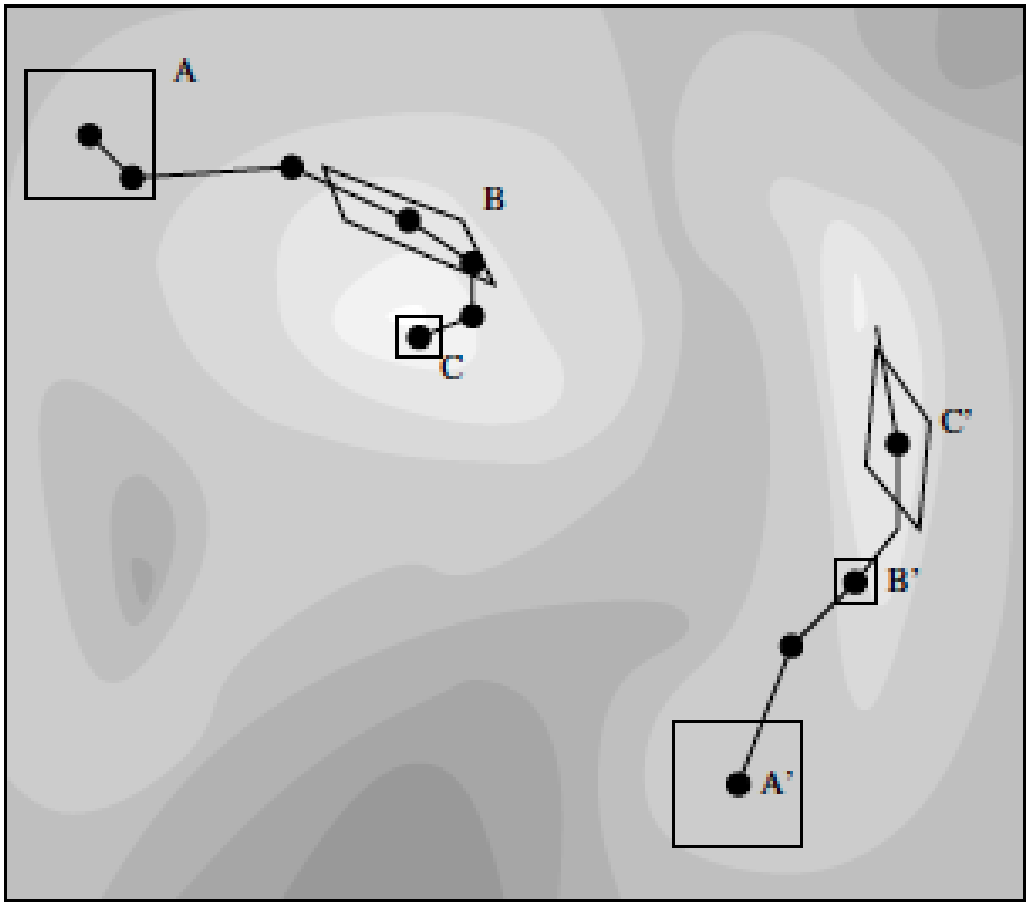
f	(input)	Function to minimize
$\boldsymbol{x}$	(input)	Initial point
$\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d$	(input)	Vectors defining search region $\mathcal{R}$ around $\boldsymbol{x}$
ρ	(input)	Box expansion factor
t	(internal)	Iteration counter
$\mathbf{P}$	(internal)	Transformation matrix
$\boldsymbol{x}, \boldsymbol{\Delta}$	(internal)	Current position, current displacement

1. **function ReactiveAffineShaker** $(f, \boldsymbol{x}, (\boldsymbol{b}_j), \rho)$
2. $t \leftarrow 0$;
3. **repeat**
4. $\boldsymbol{\Delta} \leftarrow \sum_j \text{Rand}(-1,1)\boldsymbol{b}_j$;
5. **if** $f(\boldsymbol{x} + \boldsymbol{\Delta}) < f(\boldsymbol{x})$
6. $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{\Delta}$;
7. $\mathbf{P} \leftarrow \mathbf{I} + (\rho - 1)\frac{\boldsymbol{\Delta}\boldsymbol{\Delta}^T}{\| \boldsymbol{\Delta} \|^2}$;
8. **else if** $f(\boldsymbol{x} - \boldsymbol{\Delta}) < f(\boldsymbol{x})$
9. $\boldsymbol{x} \leftarrow \boldsymbol{x} - \boldsymbol{\Delta}$;
10. $\mathbf{P} \leftarrow \mathbf{I} + (\rho - 1)\frac{\boldsymbol{\Delta}\boldsymbol{\Delta}^T}{\| \boldsymbol{\Delta} \|^2}$;
11. **else**
12. $\mathbf{P} \leftarrow \mathbf{I} + (\rho^{-1} - 1)\frac{\boldsymbol{\Delta}\boldsymbol{\Delta}^T}{\| \boldsymbol{\Delta} \|^2}$;
13. $\forall j\ \boldsymbol{b}_j \leftarrow \mathbf{P}\,\boldsymbol{b}_j$;
14. $t \leftarrow t+1$
15. **until** convergence criterion;
16. **return** $\boldsymbol{x}$;

RAS algorithm pseudo-code, comments

- Testing the function improvement on both $x+\Delta$ and $x-\Delta$ is called **double-shot** strategy
- It drastically reduces the probability of generating two consecutive unsuccessful samples
- If the double-shot strategy fails, then the transformation is applied by replacing the expansion factor ρ with its inverse ρ^{-1}
- the search speed is increased when steps are successful, reduced only if no better point is found after the double shot

Reactive affine shaker geometry



Reactive Affine Shaker geometry: two search trajectories leading to two different local minima

Repetitions for robustness and diversification

- RAS searches for local minimizers and is stopped as soon as one is found
- Even when a local minimum is found, it is generally impossible to determine whether it is global or not
- A simple way to continue the search is to **restart** from a different initial random point

The Inertial Shaker

- RAS requires matrix-vector multiplications to update the search region: it is slow if the number of dimensions is large

Solution: **Inertial shaker**:

- the search box is always identified by vectors parallel to the coordinate axes
- a **trend** direction is identified by **averaging a number of previous displacements**

Inertial Shaker, pseudo-code

f	(input)	Function to minimize
$\boldsymbol{x}$	(input)	Initial and current point
$\boldsymbol{b}$	(input)	Box defining search region $\mathcal{R}$ around $\boldsymbol{x}$
$\boldsymbol{\delta}$	(parameter)	Current displacement
amplification	(parameter)	Amplification factor for future displacements
history_depth	(parameter)	Weight decay factor for past displacement average

```
1.  function InertialShaker (f, x, b)
2.      t ← 0
3.      repeat
4.          success ← double_shot_on_all_components ( δ)
5.          if success = true
6.              x ← x + δ
7.              find_trend ( δ)
8.              if f( x + δ) < f( x)
9.                  x ← x + δ;
10.                 increase amplification and history_depth
11.             else
12.                 decrease amplification and history_depth
13.     until convergence criterion is satisfied
14.     return x;
```

Inertial Shaker, comments on the pseudo-code

- *find trend* returns a weighted average of the m_{disp} previous displacements

$$\delta_t = amplification \cdot \frac{\sum_{u=1}^{T} \delta_{t-u} e^{-\frac{u}{(history\_depth)^2}}}{\sum_{u=1}^{T} e^{-\frac{u}{(history\_depth)^2}}}$$

- *amplification* and *history depth* are defined in the algorithm
- m_{disp} is chosen in order to cut off negligible exponential weights and to keep the past history reasonably small.

GIST

- The purpose of optimization is to design automated techniques to identify inputs leading to maximum (or minimum) output values .
- Basic idea: Start from an initial value, apply small **local changes** to the inputs, test their effects. Decide whether to accept the local change or not.
- Repeat until there is progress, leading to better and better output values.

GIST (2)

- If **derivatives** are available, one can predict the effect of small local changes
- If derivatives are not available, one can test small changes directly (RAS) and keep **locally adapted models** to reduce function evaluations.
- Local adaptation occurs by **learning from the previous steps** of the search .