Function minimalization II

Marcin Chrząszcz, Danny van Dyk
mchrzasz@cern.ch,
danny.van.dyk@gmail.com

University of Zurich[UZH]

Numerical Methods,
10 October, 2016

Function minimalization

$\Rrightarrow$ Function mininalization if THE MOST important numerical technique.

$\Rrightarrow$ We will discuss finding LOCAL minimum of functions in this lecture.

$\Rrightarrow$ The numerical problem is to find the minimum one needs to evaluate the function many many times

Function minimalization

$\Rrightarrow$ Function mininalization if THE MOST important numerical technique.

$\Rrightarrow$ We will discuss finding LOCAL minimum of functions in this lecture.

$\Rrightarrow$ The numerical problem is to find the minimum one needs to evaluate the function many many times

$\Rrightarrow$ Side note: If you want to find maximum do: $f(x) = -g(x)$.

$\Rrightarrow$ How to find a minimum:

$$f(x_0 + \delta) \simeq f(x_0) + f'(x_0)\delta + \frac{1}{2} f''(x_0)\delta^2$$

$\Rrightarrow$ The first derivative $= 0$.

$\Rrightarrow$ If $|(f(x_0 + \delta) - f(x_0)| \leqslant \epsilon$ where ϵ is the accuracy we are interested in.

$\Rrightarrow$ From the above we see that:

$$|\delta| \sim \sqrt{\epsilon}$$

The minimum we are can calculate only with the $\sqrt{\epsilon}$

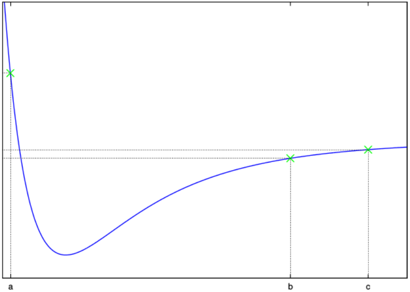
"RAW" Minimum estimation

⇛ We say that 3 points (a, b, c) are bounding the minimum of $f(x)$ if:

$$a < b < c: \ f(a) > f(b), \ f(c) > f(b) \tag{1}$$

⇛ How to find the points? The fist two we choose randomly.

⇛ Two points will show the fall of the function. We go in that direction by the same step as the distance between the first two points. If the above condition is not fulfilled we repeat the step.



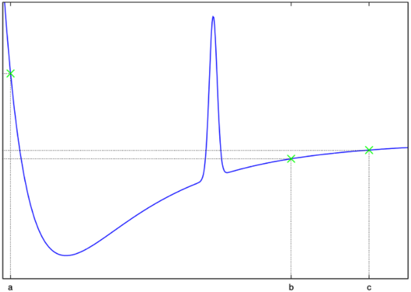
"RAW" Minimum estimation

⇛ We say that 3 points (a, b, c) are bounding the minimum of $f(x)$ if:

$$a < b < c: \;\; f(a) > f(b), \;\; f(c) > f(b) \tag{1}$$

⇛ How to find the points? The fist two we choose randomly.

⇛ Two points will show the fall of the function. We go in that direction by the same step as the distance between the first two points. If the above condition is not fulfilled we repeat the step.



"RAW" Minimum estimation

$\Rrightarrow$ If we have 3 points that fulfil 1 we choose d such that:

$$a < d < c, \;\; d \neq b$$

$\Rrightarrow$ Next steps are dependent on what value we will get.

$f(d) < f(b)$	$f(d) > f(b)$
If $d < b$ then $c = b$, $b = d$	If $d < b$ then $a = d$
If $d > b$ then $a = b$, $b = d$	If $d > b$ then $c = d$

$\Rrightarrow$ At each step we get new 3 points that fulfil the 1 so we can iterate the whole procedure. Each step will have smaller distance of the points.
$\Rrightarrow$ We and the iteration when the $|c - a| < \tau\,(|b| + |d|)$, where τ is our toleration. $\Rrightarrow$ Now the only question is how to choose the d?

"RAW" Minimum estimation

⇛ If we have 3 points that fulfil 1 we choose d such that:

$$a < d < c, \ d \neq b$$

⇛ Next steps are dependent on what value we will get.

$f(d) < f(b)$	$f(d) > f(b)$
If $d < b$ then $c = b$, $b = d$	If $d < b$ then $a = d$
If $d > b$ then $a = b$, $b = d$	If $d > b$ then $c = d$

⇛ At each step we get new 3 points that fulfil the 1 so we can iterate the whole procedure. Each step will have smaller distance of the points.

⇛ We and the iteration when the $|c - a| < \tau\,(|b| + |d|)$, where τ is our toleration. ⇛ Now the only question is how to choose the d?

Golden rule

$\Rrightarrow$ We seek for the d point in the bigger $[a,b]$, $[b,c]$.
$\Rrightarrow$ In order to minimalize the risk that the minimum will be in the smaller of the set we apply the golder rule:

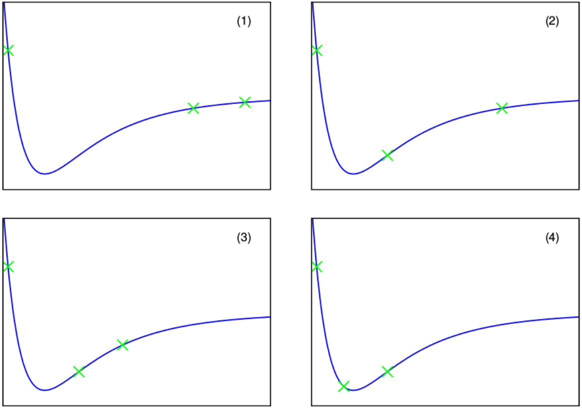
$$\frac{|b-a|}{|c-a|} = \frac{|c-b|}{|b-a|}$$

$\Rrightarrow$ Of course the initial 3 points will now have this feature, but we can obtain it fastly in the iterative procedure:

- If $|b-a| > |c-b|$, then $d = a + w|b-a|$
- If $|b-a| < |c-b|$, then $d = b + w|b-a|$

where $w = \frac{3-\sqrt{5}}{2} \simeq 0.381966...$

Golden rule



Golden rule

$\Rrightarrow$ We seek for the d point in the bigger $[a, b]$, $[b, c]$.
$\Rrightarrow$ In order to minimalize the risk that the minimum will be in the smaller of the set we apply the golder rule:

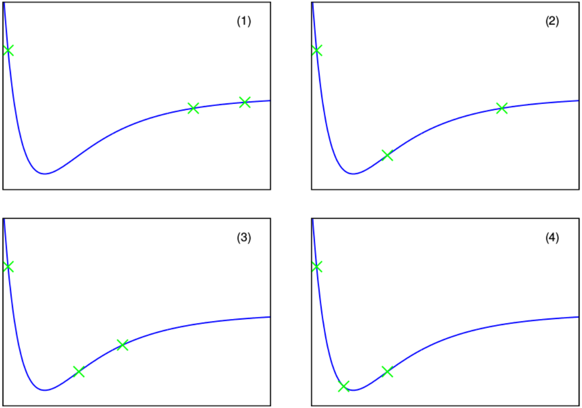
$$\frac{|b-a|}{|c-a|} = \frac{|c-b|}{|b-a|}$$

$\Rrightarrow$ Of course the initial 3 points will now have this feature, but we can obtain it fastly in the iterative procedure:

- If $|b-a| > |c-b|$, then $d = a + w|b-a|$
- If $|b-a| < |c-b|$, then $d = b + w|b-a|$

where $w = \frac{3-\sqrt{5}}{2} \simeq 0.381966...$

Golden rule - example



Brenet rule

$\Rrightarrow$ The better method then the golden rule is the parabolic interpolation.
$\Rrightarrow$ Via 3 points we draw a parabola and as a new point we choose it's minimum:

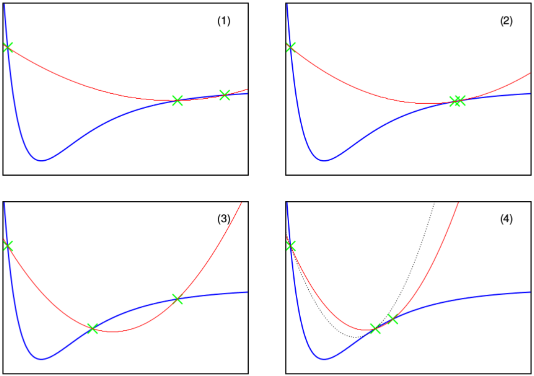
$$d = \frac{1}{2}\frac{a^2(f_c - f_b) + b^2(f_a + f_c) + c^2(f_b - f_a)}{a(f_c - f_b) + b(f_a + f_c) + c(f_b - f_a)} \tag{2}$$

$\Rrightarrow$ The justification of this method is the fact that arround the minimum the parabola is usually very good approximation.
$\Rrightarrow$ We start from 3 points that fulfil 1. The d we choose accordingly to 2.
$\Rrightarrow$ We accept the new d if:

- d is in: $a < d < c$
- The size of new set calculated accordingly to 5 is smaller then half of the set before the last iteration

$\Rrightarrow$ if they above is not true we choose d as middle point.

Hermite interpolation



⇛ The Bernet rules is consider the "standard" method for minimalizing the function.
⇛ Sometimes a better interpolation is to use the 3rd (Hermit) interpolation polynomial.

Global minimalization

Intro

$\Rrightarrow$ In many cases we know there exists one minimum, in other we are just interested in any minimum

$\Rrightarrow$ But there exists many cases we need to know the global minimum!!!

$\Rrightarrow$ All discussed till now methods are looking for the local minimum and are useless in this scenario.

$\Rrightarrow$ The discussed here global minima searches are changing year by year as this is a very fastly developing area.

$\Rrightarrow$ The global minimum finders are mostly based on some random process :)

Greedy algorithms

⇛ One can wonder why don't we just put a fine grid points and just scan every possibility for minimum?
⇛ Well this will kick you hard in many dimensions. The number of grind points go with N^n so explodes very rapidity with many dimensions.
⇛ Also starting from different points and looking for local minimums will not do you any good for the same reason.
⇛ You need to be far more intelligent in your algorithms!

Why Monte Carlo

⇛ The problem with normal methods is the fact that if they see a minimum they try to go there as fast as possible.
⇛ If you do so you will miss the global minimum.
⇛ Your algorithm needs to be able to do a step back!
⇛ Thats why MC algorithms are so good at this: The steps in better directions are favourite but also there is non-zero probability that the algorithm can go step back.

Why Monte Carlo - the algorithm

⇛ $f : \mathbb{R}^N \rightarrow \mathbb{R}$ will be the function we look for minimum.

⇛ We start from a random point $f_0 = f(x_0)$, $x_{\min} = x_0$, $f_{\min} = f_0$.

$\sigma > 0$, $T > 0$ are parameters. We do M steps:

1. We calculate the $\hat{x} = x_k + \sigma\epsilon_k$, where$\epsilon_k$ is N dim random number for Gauss distribution.
2. We calculate $\hat{f} = f(\hat{x}$
3. If $\hat{f} < f_k$ then we accept the step: $x_{k+1} = \hat{x}$, $f_{k+1} = \hat{f}$
4. If $\hat{f} < f_{\min},\ x_{\min} = x_{k+1},\ f_{\min} = \hat{f}$
5. If $\hat{f} > f_{\min}$
   - We choose z from $\mathcal{U}(0,1)$.
   - If

     $$z < \exp(-\frac{\hat{f} - f_k}{T})$$

     is true we accept new position. If not not we don't
6. We start over :)

Why Monte Carlo - the algorithm

- $\Rrightarrow$ There is no natural way to stop it.
- $\Rrightarrow$ We will do as many steps as we assumed.
- $\Rrightarrow$ Do couple of time the walk with different starting point.
- $\Rrightarrow$ The ϵ_k might not be a Gauss but some other distribution.
- $\Rrightarrow$ There are many this algorithms there: genetic algorithms, MCMC.
- $\Rrightarrow$ Current state of the art: ”Metropolis-Hastings” algorithm.
- $\Rrightarrow$ Sign up for MC course if you want to learn those :)

Backup