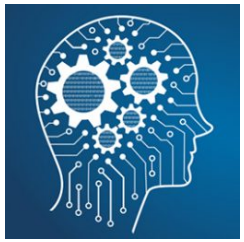
Optimization for machine learning – Part 1
Deterministic gradient algorithms

Rodolphe Le Riche[1], Didier Rullière[2]

[1] CNRS LIMOS, Mines Saint-Etienne, UCA, France
[2] Mines Saint-Etienne, CNRS LIMOS



Dec 2025
Majeure Science des données (UP3) et Master Maths en Action
Mines Saint-Etienne

Course program I

Part I. Non stochastic optimization for ML (Rodolphe)

- Introduction
  Objectives – Optimization problem formulation – Examples of optimization usages – Basic mathematical concepts for optimization
- Steepest descent algorithm
  Fixed step steepest descent algorithm – Line search – convergence
- Improved gradient based searches
  Search directions for acceleration – Making it more global: restarts – A word about constraints – Towards ML: regularized quadratic function

Course program II

Part II. Stochastic optimization for ML (Didier)

- Stochastic Approximation SA, Stochastic Gradient Descent SGD.
- Robbins-Monro

Part III. Non stochastic optimization for ML (cont., Rodolphe)

- Gradient calculation by backpropagation
- Application to neural network

Part IV. Stochastic optimization for ML (cont., Didier)

- Unknown gradient. Neural Applications and Batches.
- Kiefer-Wolfowitz – Applications to ML

Future lectures about optimization

Course program III

- Global optimization
  Using metamodels – EGO – CMAES – Simulated Annealing... ← (UP4, upcoming)

Bibliographical references for the class I

This first part of the course is based on

- [Ravikumar and Singh, 2017] : a detailed up-to-date presentation of the main convex optimization algorithms for machine learning (level end of undergraduate, bac +3)
- [Vandenberghe, 2022] : Chap. 1 ("Gradient method") provides clear information for analyzing gradient convergence (level graduate school, bac +4)
- [Sun, 2019] : review of optimization methods and good practices for tuning neural nets.

The content of these references will be simplified for this class.

Bibliographical references for the class II

Complementary recommended readings:

- [Minoux, 2008] : a classic textbook for optimization, written before the ML trend but still useful (level end of undergraduate / bac+3). The book is hard to find.
- [Bishop, 2006] : a reference book for machine learning with some pages on optimization (level end of undergraduate / bac+3)
- [Schmidt et al., 2007] : L1 regularization techniques (research article)

Optimization = a quantitative formulation of decision

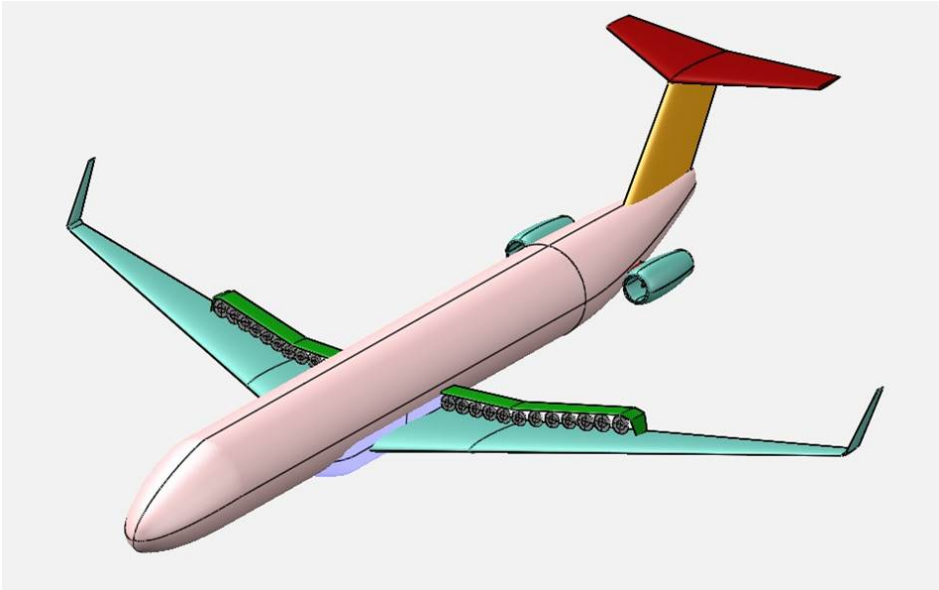
Optimization is a[1] way of mathematically modeling decision.

$$\min_{x \in \mathcal{S}} f(x)$$

- x vector of decision parameters (variables) : dimensions, investment, tuning of a machine / program, ...
- $f(x)$: decision cost x
- $\mathcal{S}$: set of possible values for x, search space

[1] non unique, incomplete when considering human beings or life

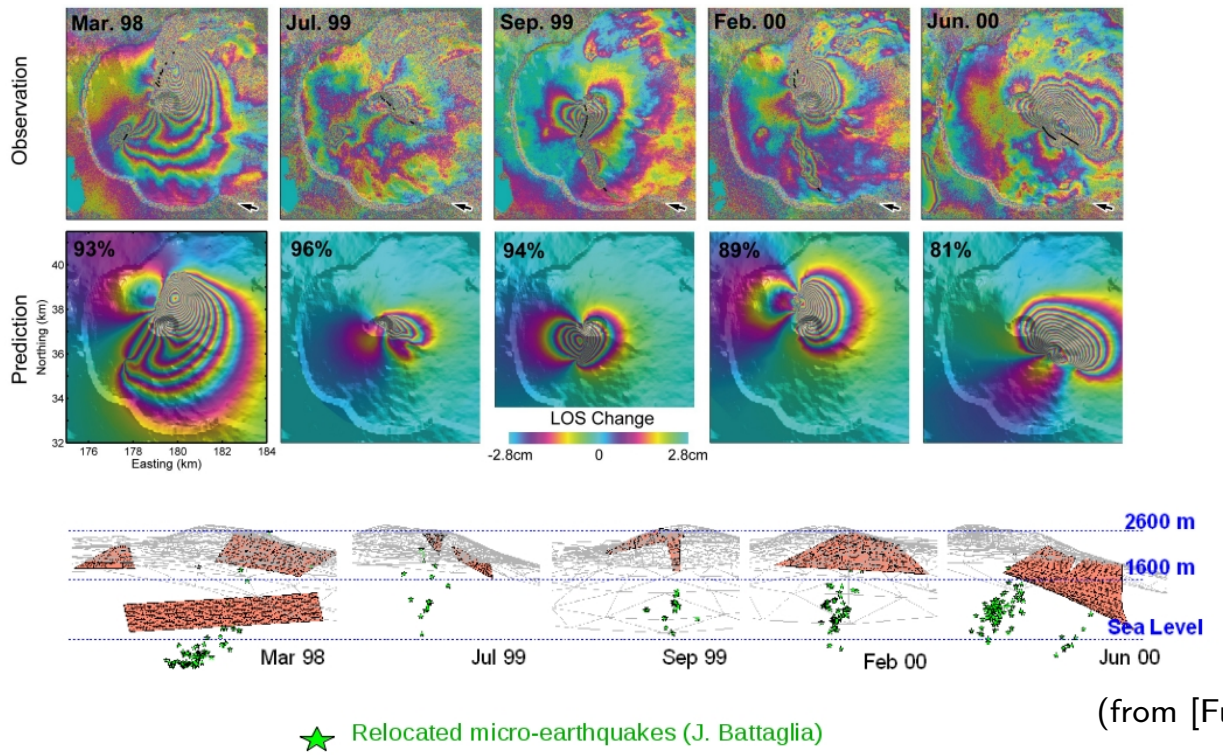
Optimization example: design



(from [Sgueglia et al., 2018])

x = aircraft parameters (here distributed electrical propulsion)
$f() = -1\times$ performance metric (aggregation of $-1\times$ range, cost, take-off length, ...)
At the minimum, the design is "optimal".

Optimization example: model identification



(from [Fukushima et al., 2010])

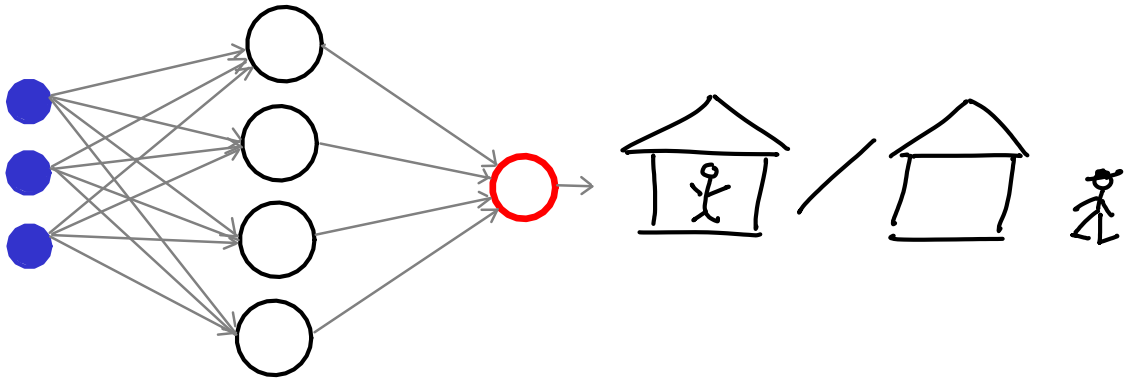
x = dike position, geometry, internal pressure

$f()$ = distance between measures (from RADARSAT-1 satellite) and model (boundary elements, non trivial computation)

At the minimum, the model best matches measurements and should correspond to the underground phenomenon.

Optimization example: neural net classification

Predict if a person stays at home or goes out based on longitude, latitude and temperature = a 2 classes classification problem.



x = neural network (NN) weights and biases

$f()$ = an error of the NN predictions (a cross-entropy error):

- e entries: e_1 longitude, e_2 latitude, e_3 temperature
- $t = 1$ if person stays, $t = 0$ otherwise
- Observed data set: $(e^i, t^i),\ i = 1, \ldots, N$
- $y(e; x)$: output of the NN, the probability that $t(e) = 1$
- $f(x) = -\sum_{i=1}^{N} \{t^i \log(y(e^i; x)) + (1 - t^i) \log(1 - y(e^i; x))\}$

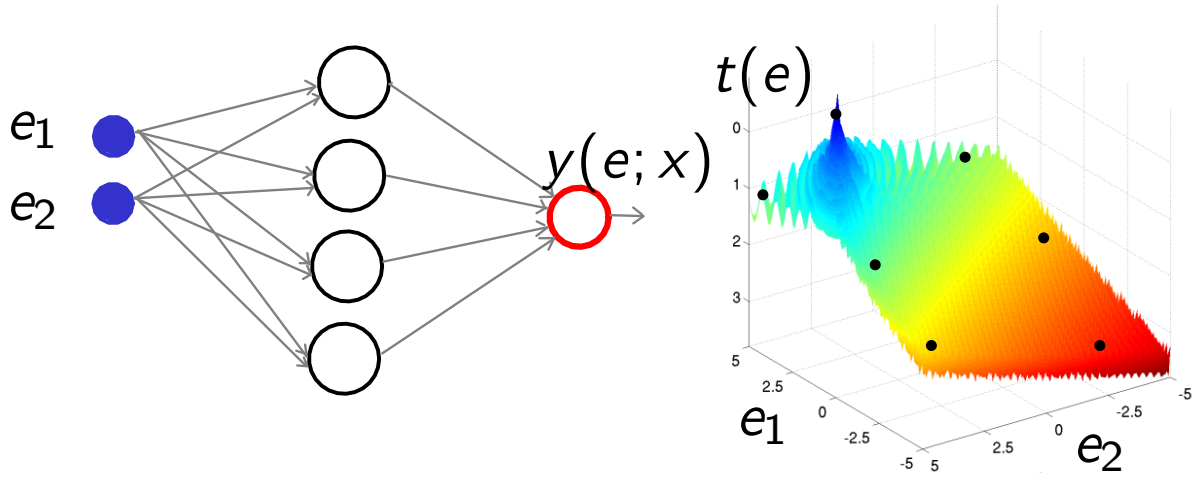
(a word on the classification cross-entropy error)

- View the relationship between the entry e and the class t as probabilistic (generalizes deterministic functions): $t(e)$ is a Bernoulli variable with a given probability that $t(e) = 1$
- The NN models this probability: $y(e; x)$ is the probability that $t(e) = 1$, $1 - y(e; x)$ is the proba that $t(e) = 0$, $0 < y(e; x) \leq 1$.
- The probability of t knowing e can be written $y(e; x)^t \times (1 - y(e; x))^{1-t}$
- The likelihood of the N i.i.d observations is $\prod_{i=1}^{N} \left[y(e^i; x)^{t^i} \times (1 - y(e^i; x))^{1-t^i} \right]$, to be maximized
- The likelihood is turned into an error, to be minimized, by taking $-\log(\text{likelihood})$,

$$f(x) = -\sum_{i=1}^{N} \{ t^i \log(y(e^i; x)) + (1 - t^i) \log(1 - y(e^i; x)) \}$$

Optimization example: neural net regression

learn a function from a discrete limited set of observations



x = neural network (NN) weights and biases

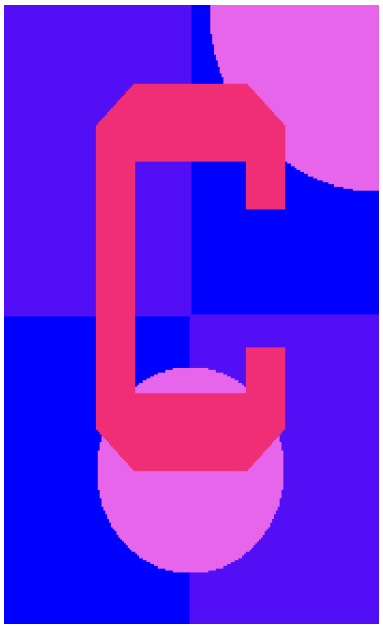
$f()$ = an error of the NN predictions (sum-of-squares error):

- e entries, $t(e)$ target function to learn
- observed data set, "$\bullet$" : $(e^i, t^i),\ i = 1, \ldots, N$
- $y(e;x)$: output of the NN, the expected value of $t(e)$
- $f(x) = 1/2 \sum_{i=1}^{N} (t^i - y(e^i;x))^2$

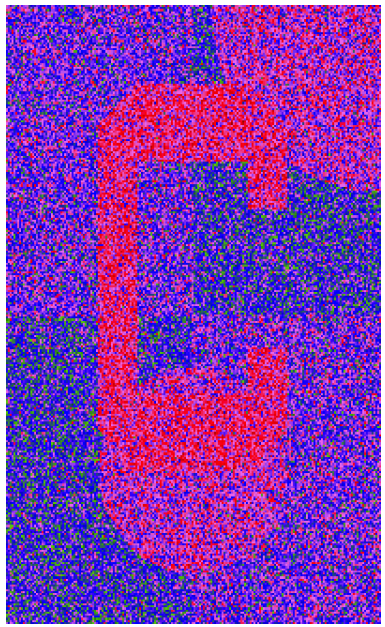
Optimization example: image denoising

$$\min_x f(x) \quad , \quad f(x) = \frac{1}{2} \sum_{i=1}^{N_{\text{pixels}}} (y_i - x_i)^2 + \lambda \sum_{i=1}^{N_{\text{pixels}}} \sum_{j \text{ near } i} |x_i - x_j|$$

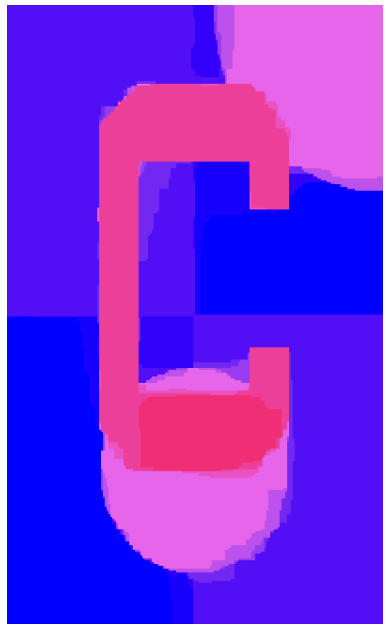
$\lambda \geq 0$ regularization constant



target image



noisy (observed)
$= y_i$'s



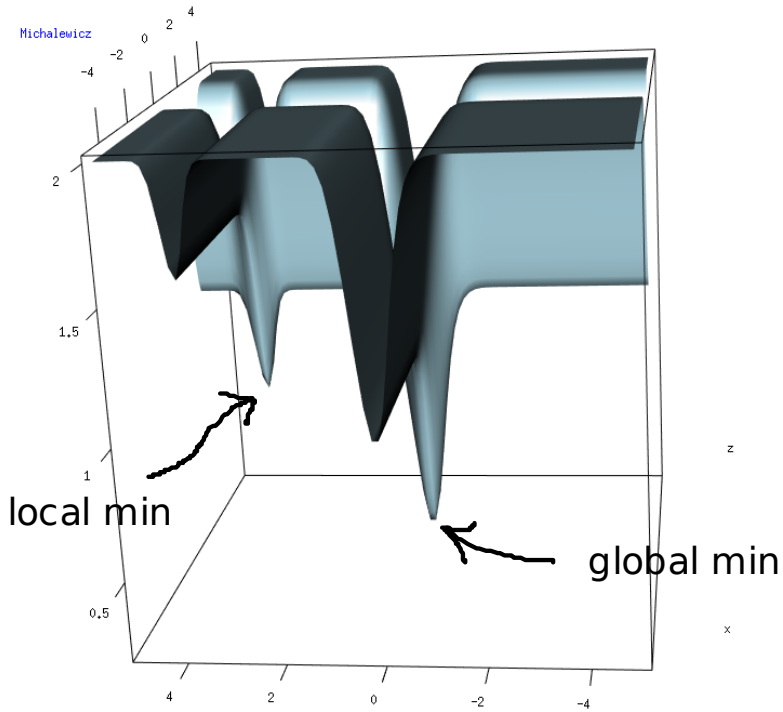
denoised (optimized)
$= x^\star$

(from [Ravikumar and Singh, 2017])

Basic mathematical concepts for optimization

Local versus global optimum

$$\min_{x \in \mathcal{S} \subset \mathbb{R}^D} f(x)$$

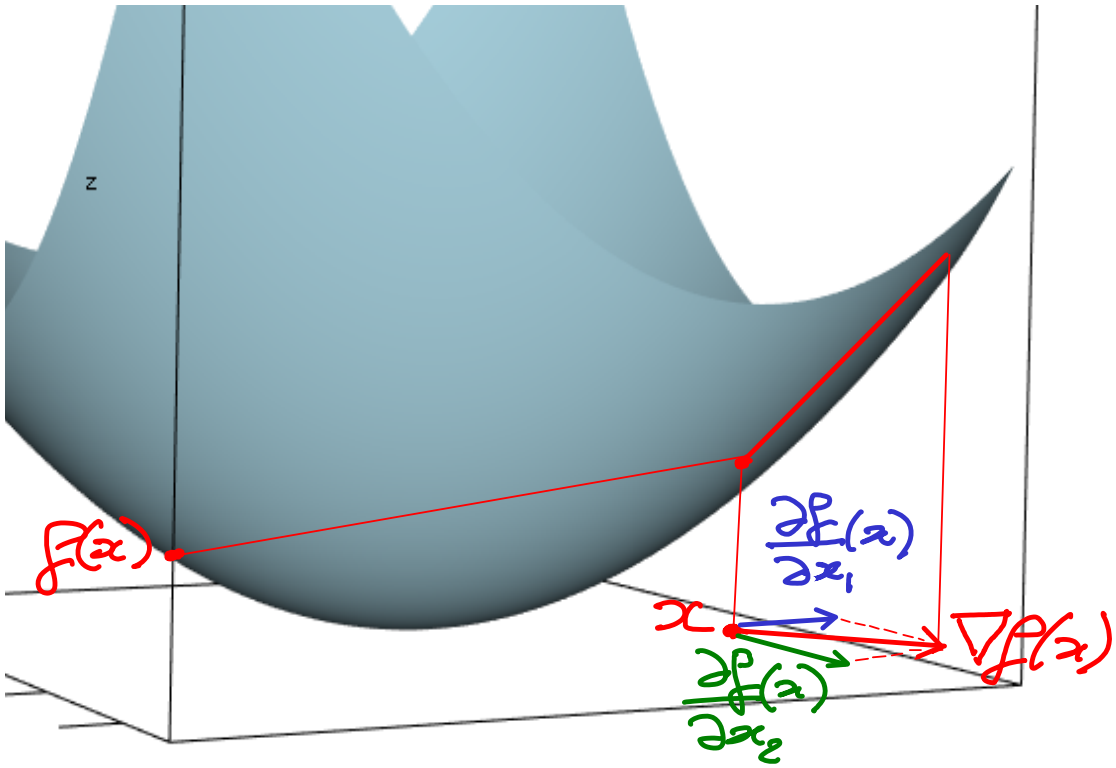


R code to generate such a 3D plot given during the practicum, cf. `TP_gradient.Rmd`

Gradient of a function

Gradient of a function = direction of steepest ascent = vector of partial derivatives

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(x) \\ \dots \\ \frac{\partial f}{\partial x_D}(x) \end{pmatrix}$$



Numerical approximation of the gradient

By forward finite differences

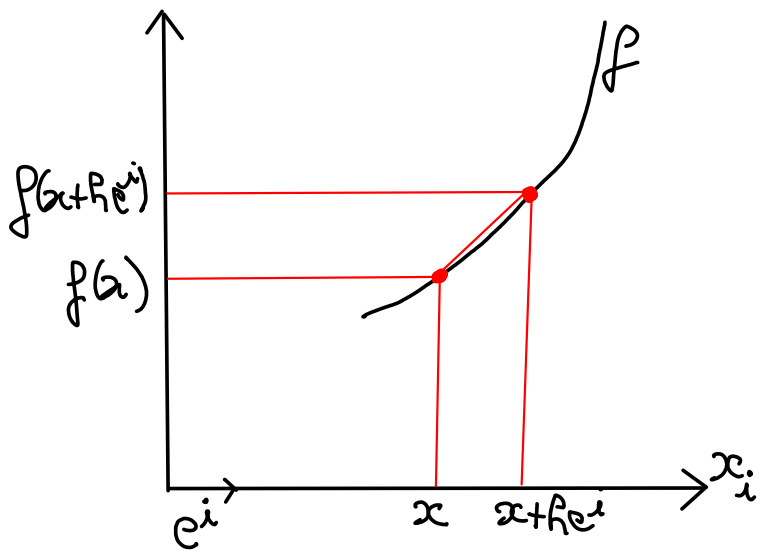
$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x + he^i) - f(x)}{h}$$

Proof: by Taylor,

$f(x + he^i) = f(x) + he^{i\top}.\nabla f(x) + h^2/2 e^{i\top}\nabla^2 f(x + \rho he^i)e^i$, $\rho \in [0, 1]$

$\frac{\partial f(x)}{\partial x_i} = \frac{f(x+he^i)-f(x)}{h} - h/2 e^{i\top}\nabla^2 f(x + \rho he^i)e^i$

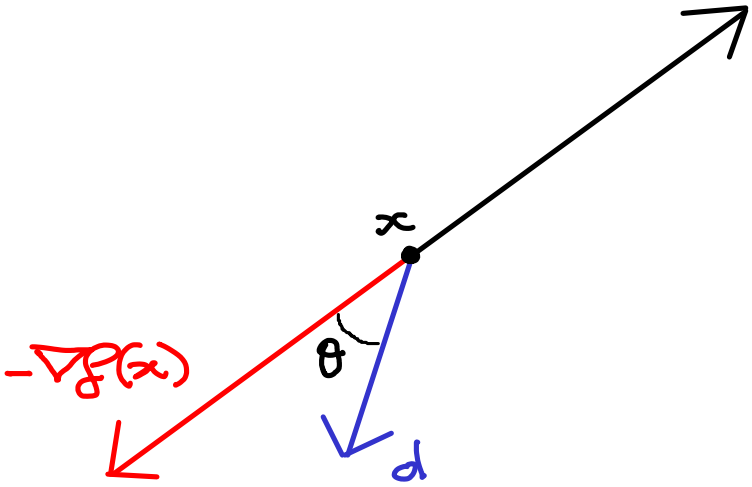
and make h very small $\square$



Other (better but more difficult to implement) schemes: central differences, automatic differentiation (e.g., in TensorFlow or PyTorch), (semi-)analytic differentiation (e.g., backpropagation in NN).

Descent direction

A search direction d which makes an acute angle with $-\nabla f(x)$ is a descent direction, i.e., for a small enough step f is guaranteed to decrease!

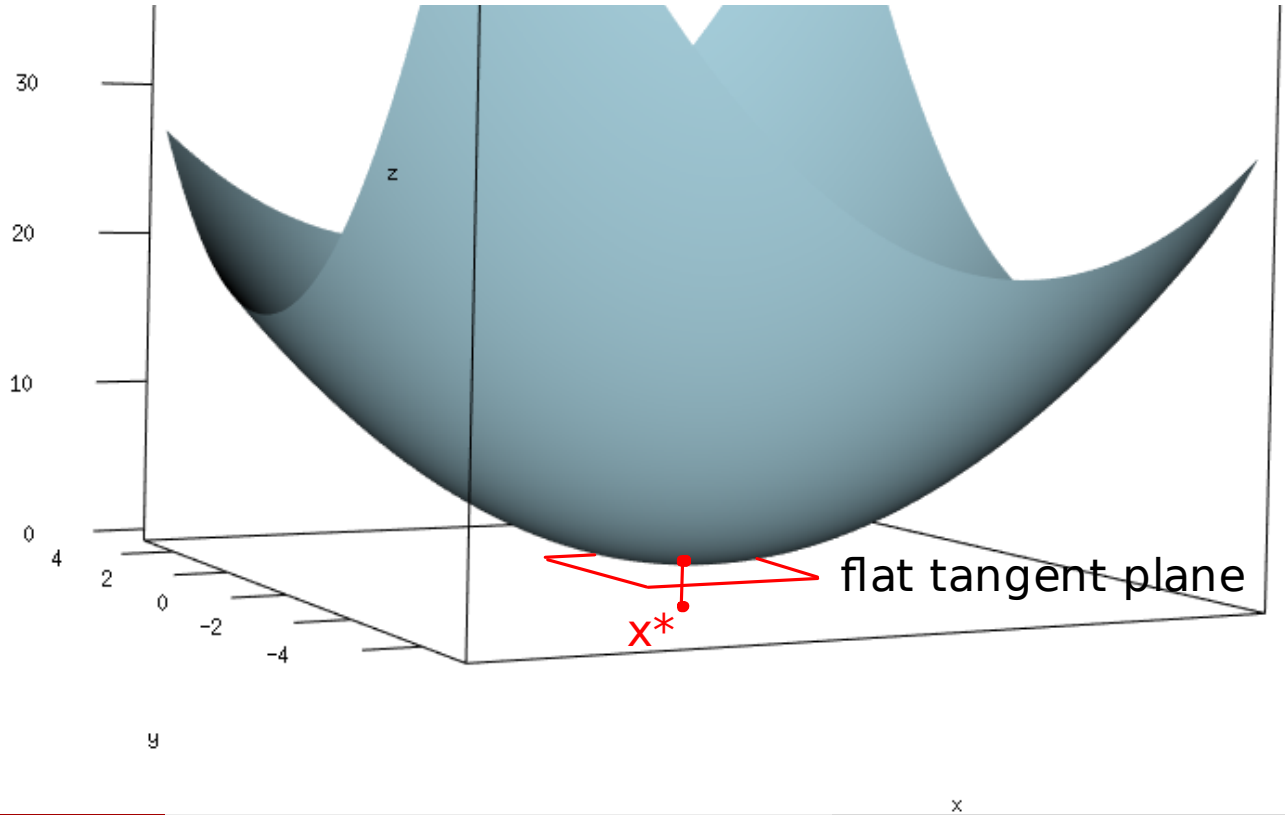


Proof: by Taylor, $\forall \alpha \geq 0,\ \exists \epsilon \in [0,1]$ such that
$f(x+\alpha d) = f(x) + \alpha d^\top . \nabla f(x) + \frac{\alpha^2}{2} d^\top \nabla^2 f(x+\alpha\epsilon d) d$
$\lim_{\alpha \to 0^+} \frac{f(x+\alpha d)-f(x)}{\alpha} = d^\top . \nabla f(x) = -1 \times \|\nabla f(x)\| \cos(d, -\nabla f(x))$
is negative if the cosine is positive $\square$

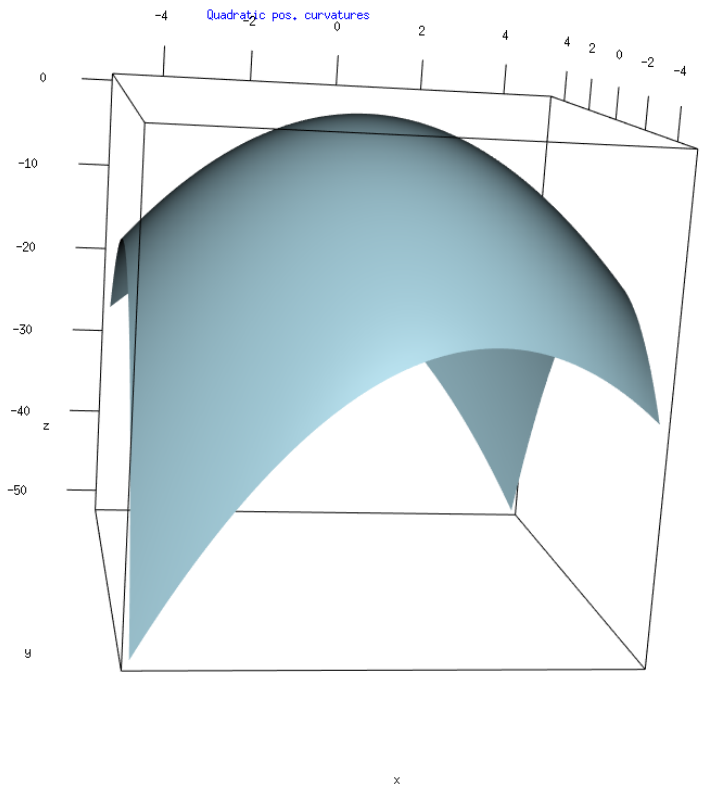
Necessary optimality condition (1)

A necessary condition for a differentiable function to have a minimum at $x^\star$ is that it is flat at this point, i.e., its gradient is null

$$x^\star \in \arg\min_{x \in \mathcal{S}} f(x) \Rightarrow \nabla f(x^\star) = 0$$

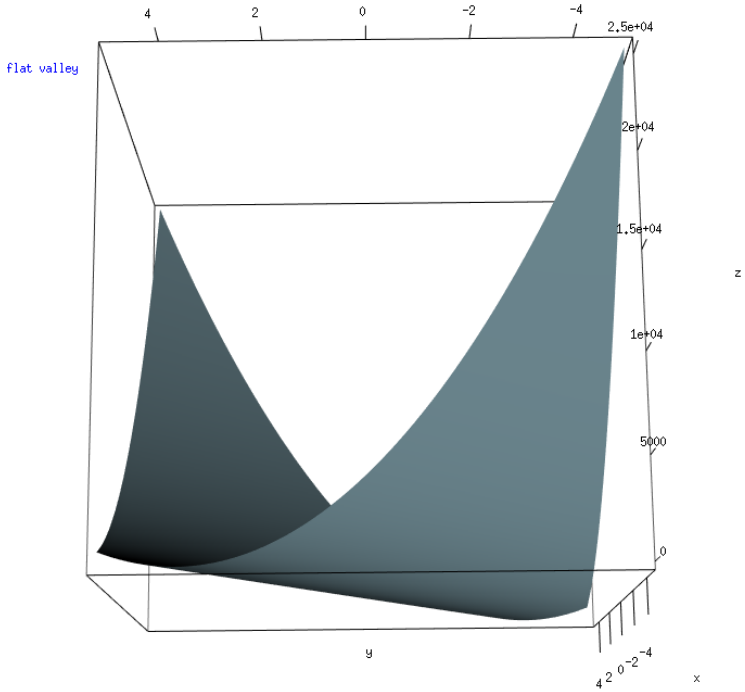


Necessary optimality condition (2)



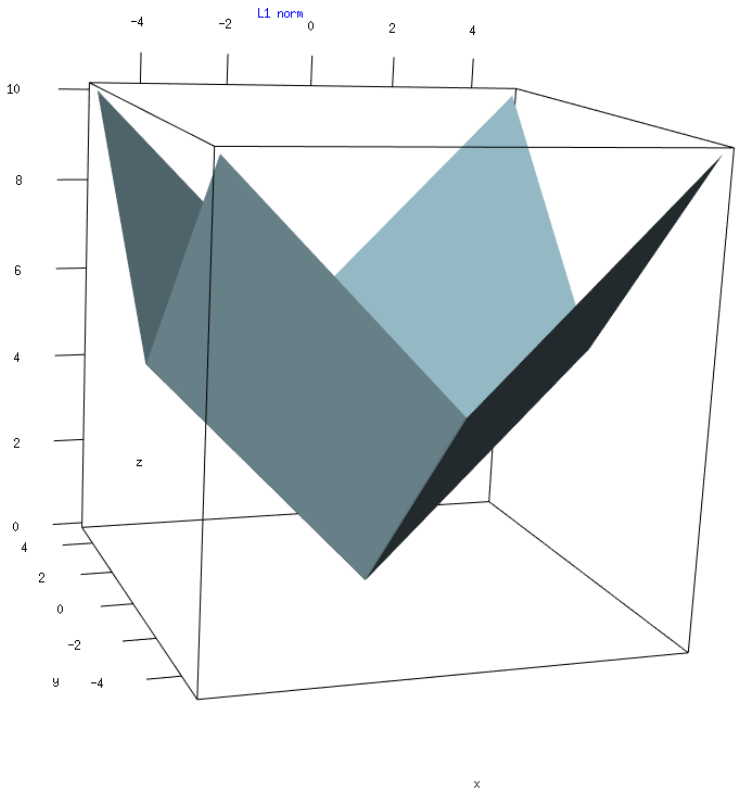
necessary is not sufficient (works with a max)

Necessary optimality condition (3)



$\nabla f(x^\star) = 0$ does not make $x^\star$ unique (flat valley)

Necessary optimality condition (4)



$\nabla f()$ not defined everywhere, example with L1 norm $= \sum_i^D |x_i|$

Course outline

Optimization for machine learning

Optimizers as iterative algorithms

$$\text{We look for} \quad x^\star \in \arg\min_{x \in \mathcal{S}} f(x) \quad , \quad \mathcal{S} = \mathbb{R}^D$$

- Except for special cases (e.g., convex quadratic problems), the solution is not obtained analytically through the optimality conditions ($\nabla f(x^\star) = 0 +$ higher order conditions).
- We typically use iterative algorithms: x^{i+1} depends on previous iterates, $x^1, \ldots, x^i$ and their f's.
- Often calculating $f(x^i)$ takes more computation than the optimization algorithm itself.
- Qualities of an optimizer: robustness, speed of convergence. Often have to strike a compromise between them.

Fixed step steepest descent algorithm (1)

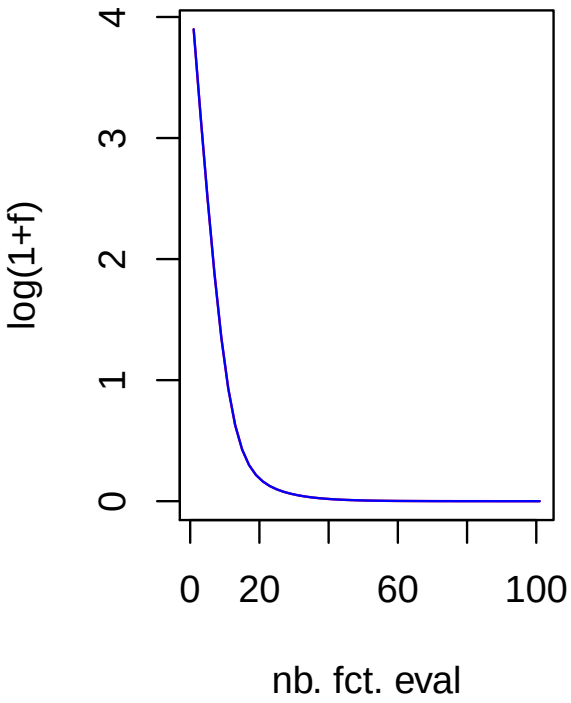
Repeat steps along the steepest descent direction, $-\nabla f(x^t)$ [Cauchy et al., 1847].
The size of the steps is proportional to the gradient norm.

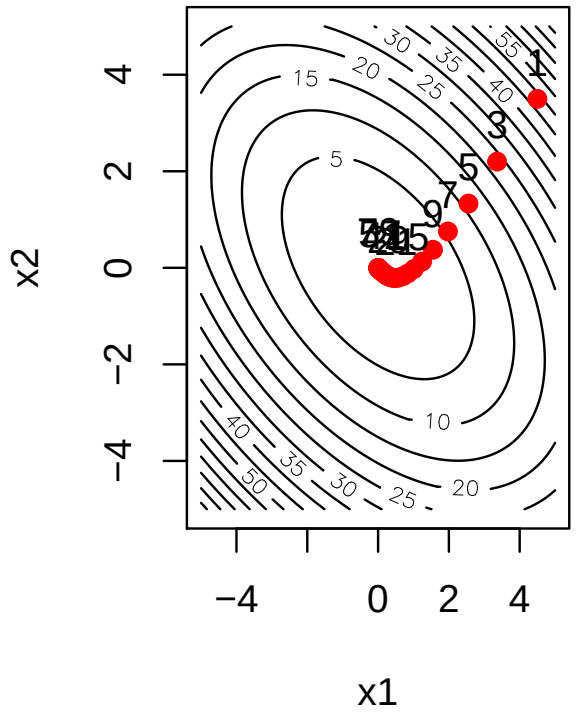
Require: $f()$, $\bar{\alpha} \in]0, 1]$, x^1, ϵ^{step}, ϵ^{grad}, $i^{\max}$
  $i \leftarrow 0$, $f^{\text{best so far}} \leftarrow$ `max_double`
  repeat
    $i \leftarrow i + 1$
    calculate $f(x^i)$ and $\nabla f(x^i)$
    if $f(x^i) < f^{\text{best so far}}$ **then**
      update $x^{\text{best so far}}$ and $f^{\text{best so far}}$ with current iterate
    end if
    direction: $d^i = -\nabla f(x^i)/\|\nabla f(x^i)\|$
    step: $x^{i+1} = x^i + \bar{\alpha}\|\nabla f(x^i)\| d^i$
  until $i > i^{\max}$ **or** $\|x^i - x^{i-1}\|/\sqrt{D} \leq \epsilon^{\text{step}}$ **or** $\|\nabla f(x^i)\|/\sqrt{D} \leq \epsilon^{\text{grad}}$

Fixed step steepest descent algorithm (2)

- The choice of the gradient scale factor $\bar{\alpha}$ is critical : the steeper the function, the smaller $\bar{\alpha}$. Default value $= 0.1$
- The true code (cf. `gradient_descent.py`) is much longer and filled with instructions for reporting the points visited and satisfying bound constraints.

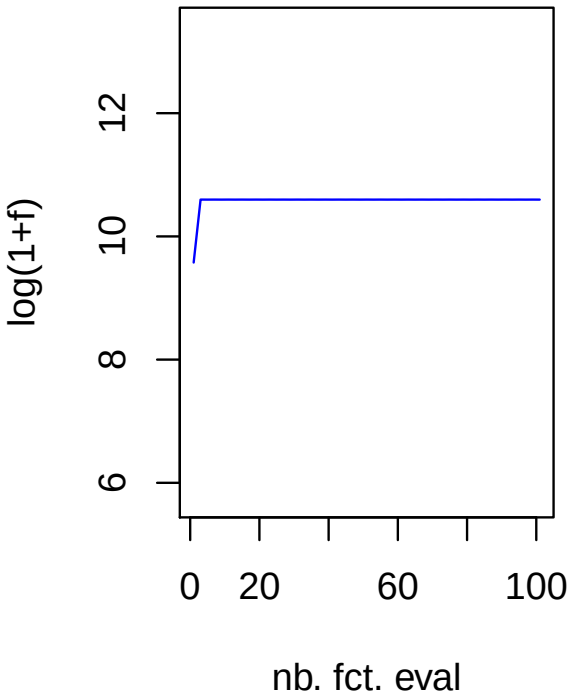
$f(x) = 1/2 x^\top H x$, H positive definite

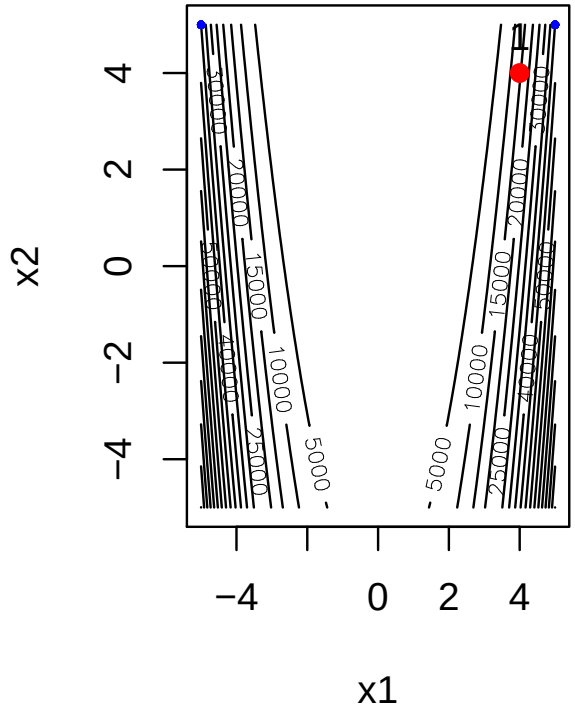




Fixed step steepest descent algorithm (3)

$\bar{\alpha} = 0.1$ on $f(x) =$ Rosenbrock (banana shaped) function in $D = 2$ dimensions, example of ping-pong phenomenon (too large steps) due to large gradients and bounded domain:
if $x_i < LB_i,\ x_i \leftarrow LB_i$, if $x_i > UB_i, x_i \leftarrow UB_i,\ i = 1, \ldots, D$





$$x^\star = (1, 1)\ ,\ f(x^\star) = 0$$

Descent with line search

At each iteration, search for the best step size in the descent direction d^i (which for now is $-\nabla f(x^i)/\|\nabla f(x^i)\|$ but it is general). Same algorithm as before, just change the **step** instruction:

Require: ...
  initializations but no α now ...
  repeat
    increment i, calculate $f(x^i)$ and $\nabla f(x^i)$ ...
    direction: $d^i = -\nabla f(x^i)/\|\nabla f(x^i)\|$ or any other **descent** direction
    step: $\alpha^i = \arg\min_{\alpha>0} f(x^i + \alpha d^i)$
    $x^{i+1} = x^i + \alpha^i d^i$
  until stopping criteria
  return best so far

Non descent direction:Prove that if d^i is not a descent direction, $-d^i$ is.

Approximate line search (1)

Notation: during line search i,

$$\begin{aligned} x &= x^i + \alpha d^i \\ f(\alpha) &= f(x^i + \alpha d^i) \\ \frac{df(0)}{d\alpha} &= \sum_{j=1}^{D} \frac{\partial f(x^i)}{\partial x_j} \frac{\partial x_j}{\partial \alpha} = \sum_{j=1}^{D} \frac{\partial f(x^i)}{\partial x_j} d_j^i = \nabla f(x^i)^\top . d^i \end{aligned}$$

In practice, perfectly optimizing for α^i is too expensive and not useful $\Rightarrow$ approximate the line search by a sufficient decrease condition:

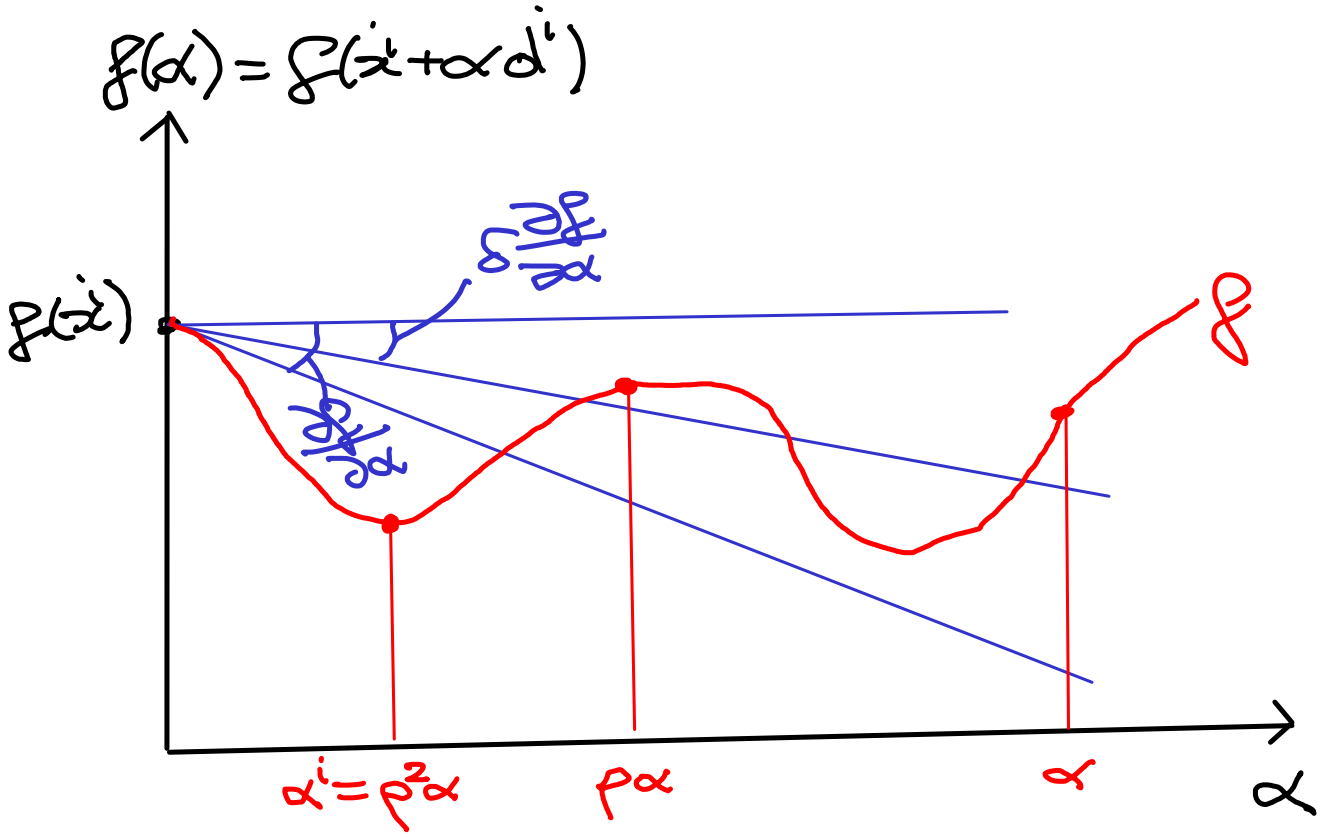
$$\text{find } \alpha^i \text{ such that } f(x^i + \alpha^i d^i) < f(x^i) + \delta \alpha^i \nabla f(x^i)^\top . d^i$$

where $\delta \in [0, 1]$, i.e., achieve a δ proportion of the progress promised by order 1 Taylor expansion.

Approximate line search (2)

Sufficient decrease condition rewritten with line search notation:

$$\text{find } \alpha^i \text{ such that } f(\alpha^i) < f(x^i) + \delta\alpha^i \frac{df(0)}{d\alpha}$$



$$\rho \in]0, 1[$$

Approximate line search (3)

At iteration i:

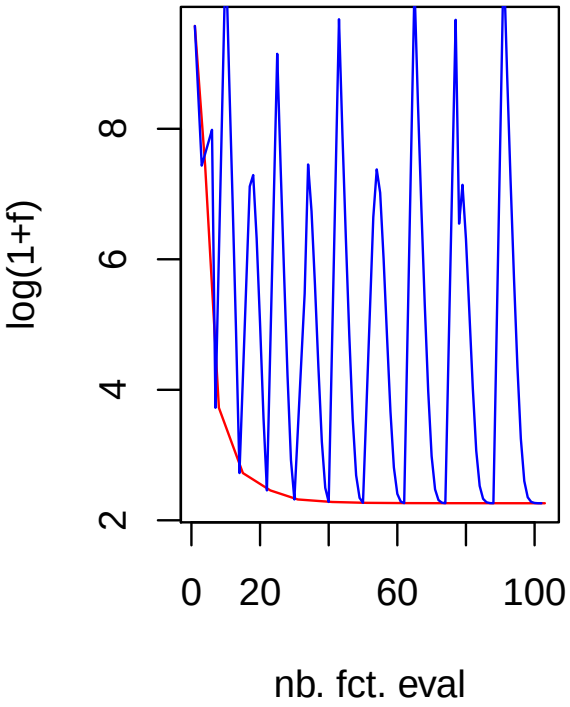
Backtracking line search (Armijo)

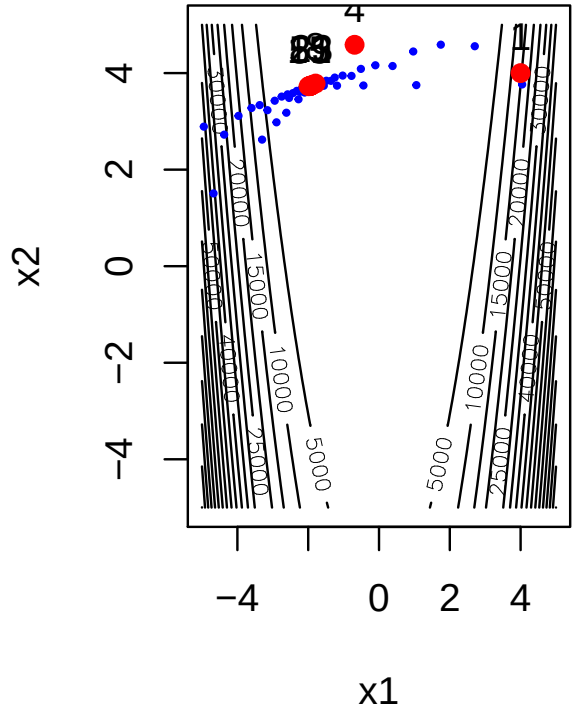
Require: d a descent direction, x^i, $\delta \in [0, 1]$, $\rho \in]0, 1[$, $C > 0$
(defaults: $\delta = 0.1$, $\rho = 0.5$, $C = 1$)
initialize step size: $\alpha = \max(C \times \|\nabla f(x^i)\|\ ,\ \sqrt{D}/100)$
while $f(x^i + \alpha d^i) \geq f(x) + \delta\alpha\nabla f(x^i)^\top . d$ **do**
decrease step size: $\alpha \leftarrow \rho \times \alpha$
end while
return $\alpha^i \leftarrow \alpha$

From now on, use line search, and the number of calls to f is no longer equal to the iteration number since many function calls can be done during a line search within a single iteration.

Approximate line search (4)

Look at what line search does to $f(x) =$ Rosenbrock where fixed step size went out of the domain





Better, but not perfect: oscillations make progress very slow.

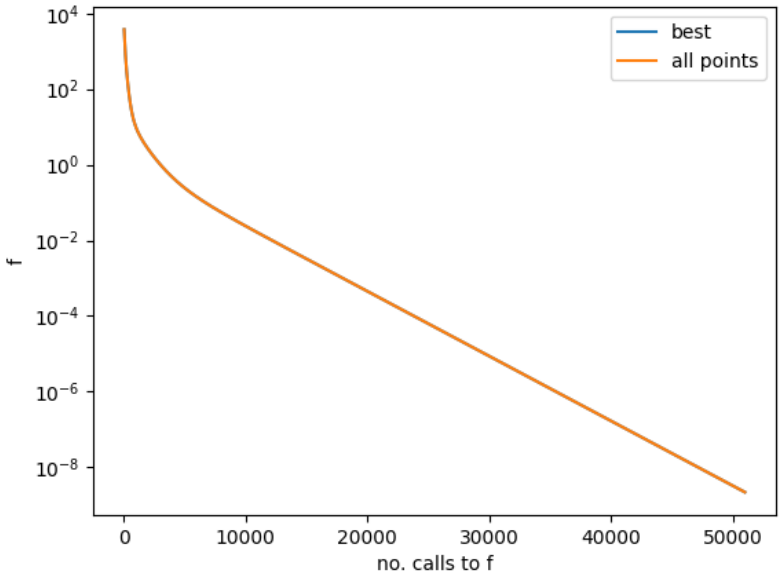
Gradient convergence : summary I

[Vandenberghe, 2022, Ravikumar and Singh, 2017, Li, 2019]: on convex and L-strongly smooth functions, gradient search with constant and sufficiently small scale factor $\bar{\alpha}$ achieves after T iterations

$$f(x^T) - f(x^\star) \leq \frac{\|x^0 - x^\star\|}{2\bar{\alpha}\, T} \tag{1}$$

i.e., $\mathcal{O}(1/T)$ convergence. To have $f(x^T) - f(x^\star) < \varepsilon$, $T > \mathcal{O}(1/\varepsilon)$

Gradient convergence : summary II



Experiment: $f(x) = \frac{1}{2}x^\top Hx$ in $D = 50$ dimensions, $H > 0$, not aligned with the axes, condition number $= 50$.

Convergence is faster than Eq.(1) $\Leftarrow$ the function is L-strongly smooth but also m-strongly convex, for which one has

$$\|x^T - x^\star\|^2 \leq c^T \|x^0 - x^\star\|^2$$

$$\text{where } 0 < c < 1\ ,\ c = 1 - \bar{\alpha}\bullet\ ,\ \bullet = m \text{ or } \frac{2mL}{m+L} \quad (2)$$

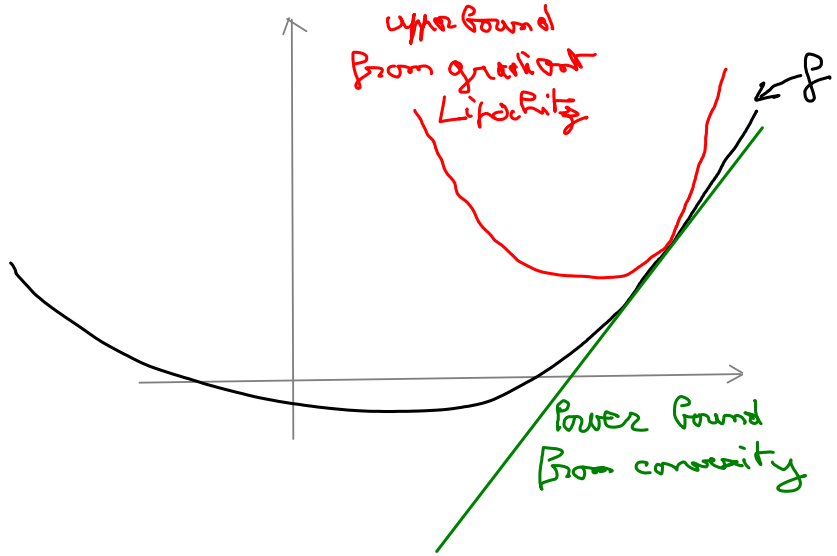
We return to these results next.

Analysis : always improving I

Speepest descent with constant step size

- One iteration of the algorithm : $y = x - \bar{\alpha}\nabla f(x)$
  $(x \equiv x^{t-1},\ y \equiv x^t)$
- L-strongly smooth function

$$f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}\|y - x\|^2 \quad (3)$$



(L-strongly smooth $\equiv$ gradient Lipschitz in convex domain, $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$), cf. Appendix in slide 61)

Analysis : always improving II

- $y = x - \bar{\alpha}\nabla f(x)$ (next iterate)
- Together,

$$f(y) \leq f(x) - \left(1 - \frac{L\bar{\alpha}}{2}\right) \bar{\alpha}\|\nabla f(x)\|^2$$

- Take $0 < \bar{\alpha} < 1/L$,

$$f(y) \leq f(x) - \frac{\bar{\alpha}}{2}\|\nabla f(x)\|^2$$

$\Rightarrow$ with $\bar{\alpha}$ small enough, f is guaranteed to decrease at the next step

Analysis : always improving III

- Convexity : $f(x) \leq f(x^\star) + \nabla f(x)^\top (x - x^\star)$

$$
\begin{aligned}
f(y) &\leq f(x^\star) + \nabla f(x)^\top (x - x^\star) - \frac{\bar{\alpha}}{2} \|\nabla f(x)\|^2 \\
f(y) - f(x^\star) &\leq \left(2\bar{\alpha} \nabla f(x)^\top (x - x^\star) - \bar{\alpha}^2 \|\nabla f(x)\|^2\right) / (2\bar{\alpha}) \\
f(y) - f(x^\star) &\leq \left(\|x - x^\star\|^2 - \|x - \bar{\alpha}\nabla f(x) - x^\star\|^2\right) / (2\bar{\alpha}) \\
0 \leq f(y) - f(x^\star) &\leq \left(\|x - x^\star\|^2 - \|y - x^\star\|^2\right) / (2\bar{\alpha})
\end{aligned}
$$

$\Rightarrow$ progress is always made towards $x^\star$

Analysis : convergence speed I

- Sum from $t = 1$ to T, going back to the superscript notation,

$$\begin{aligned}\sum_{t=1}^{T} \left(f(x^t) - f(x^\star)\right) &\leq \sum_{t=1}^{T} \left(\|x^{t-1} - x^\star\|^2 - \|x^t - x^\star\|^2\right)/(2\bar{\alpha}) \\ &= \|x^0 - x^\star\|^2/(2\bar{\alpha}) - \|x^T - x^\star\|^2/(2\bar{\alpha}) \\ &\leq \|x^0 - x^\star\|^2/(2\bar{\alpha})\end{aligned}$$

- In addition, because the function is decreasing, $\sum_{t=1}^{T} \left(f(x^t) - f(x^\star)\right) \geq T\left(f(x^T) - f(x^\star)\right)$, thus

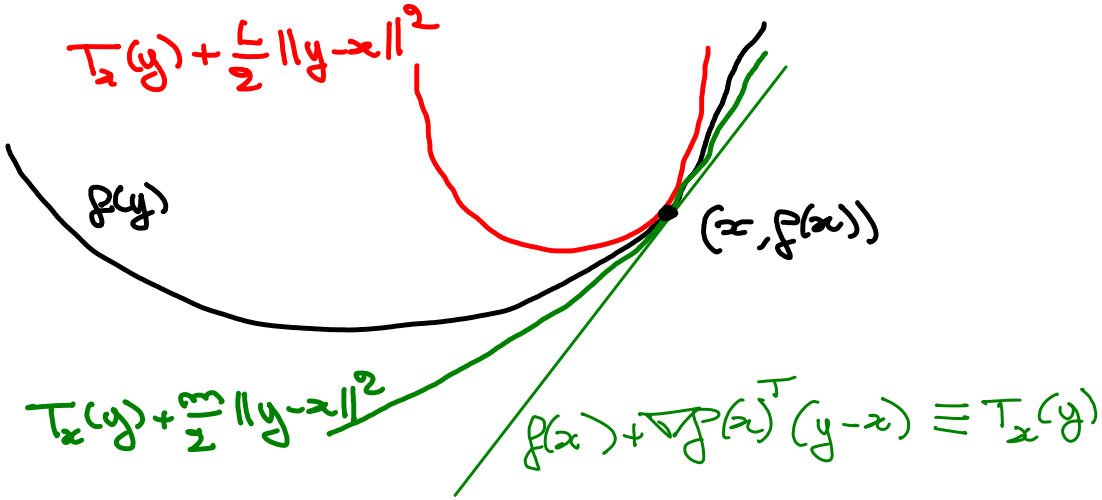
$$f(x^T) - f(x^\star) \leq \frac{\|x^0 - x^\star\|}{2\bar{\alpha}T}$$

which proves (1).
For a faster convergence: increase $\bar{\alpha}$ (but $\bar{\alpha} \leq 1/L$) or start closer to solution.

Analysis : convergence speed II

- Convergence speed in the x-space requires a m-strong convexity assumption

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{m}{2}\|y - x\|^2 \quad , \quad \forall x, y \in \mathcal{S}$$



Analysis : convergence speed III

$$\|x^T - x^\star\|^2 \leq c^T \|x^0 - x^\star\|^2$$

where $0 < c < 1$, $c = 1 - \bar{\alpha}\bullet$

(proof when $\bullet = \frac{2mL}{m+L}$ in [Vandenberghe, 2022], Chapter on Gradient Descent, and more compact proof that $\bullet = m$ in Convergence Rates slides of Stochastic Gradient part of the course)

log-convergence in $1/\varepsilon$: Prove that
$\|x^T - x^\star\|^2 \leq \varepsilon \;\Rightarrow\; T \geq \mathcal{O}(\log(1/\varepsilon))$

Analysis: oscillations

Perfect line search solves

$$\alpha^i = \arg\min_{\alpha>0} f(\alpha) \quad \text{where} \quad f(\alpha) = f(x^i + \alpha d^i)$$

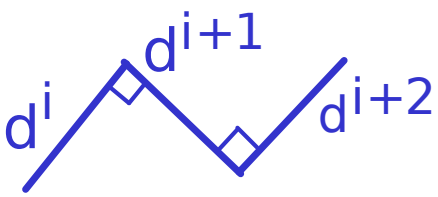
Necessary conditions of optimal step size:

$$\frac{df(\alpha^i)}{d\alpha} = \sum_{j=1}^{d} \frac{\partial f(x^i + \alpha^i d^i)}{\partial x_j} \frac{\partial x_j}{\partial \alpha} = \nabla f(x^{i+1})^\top . d^i = 0$$

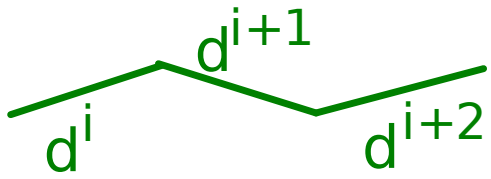
If the direction is the gradient,

$$-{d^{i+1}}^\top . d^i = 0 \quad \text{i.e. } d^{i+1} \text{ and } d^i \text{ perpendicular}$$

gradient does



less oscillations seems better



Course outline

Optimization for machine learning

Gradient with momentum I

Recall fixed step gradient descent,

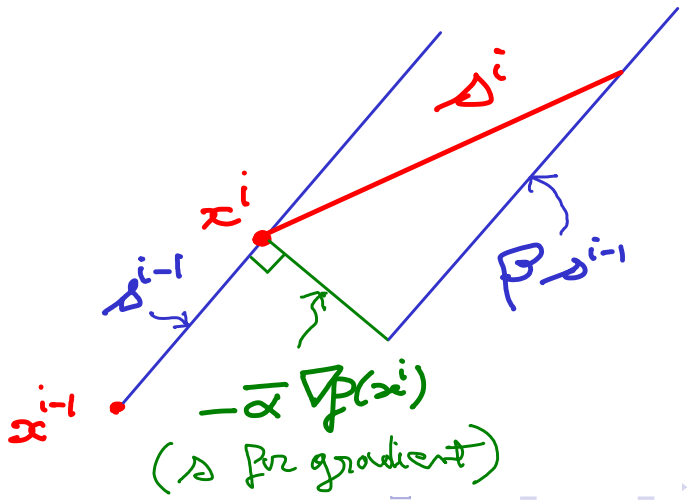
$$x^{i+1} = x^i - \bar{\alpha}\nabla f(x^i) \quad \text{i.e.,} \quad s^i = -\bar{\alpha}\nabla f(x^i)$$

s^i, the step. Should not be mistaken for $x^{i+1} - x^i$ where line search and bounds corrections have occured.
Introduce a momentum (a memory) in the search step [Polyak, 1964],

$$s^i = -\bar{\alpha}\nabla f(x^i) + \beta s^{i-1}$$

where $\beta = 0.9$. (alternatively, for iteration varying momentum, $\beta^i = (i-2)/(i+1)$)

This should contribute to avoid the oscillations occuring at the botton of valleys.



Gradient with momentum II

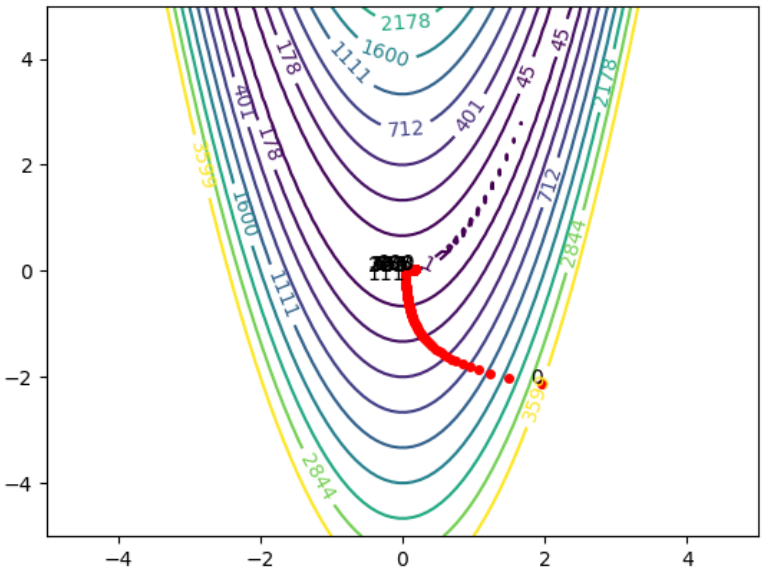
s^i still a descent direction.

Momentum direction descends: Assuming perfect line searches, prove that s^i with momentum is a descent direction.

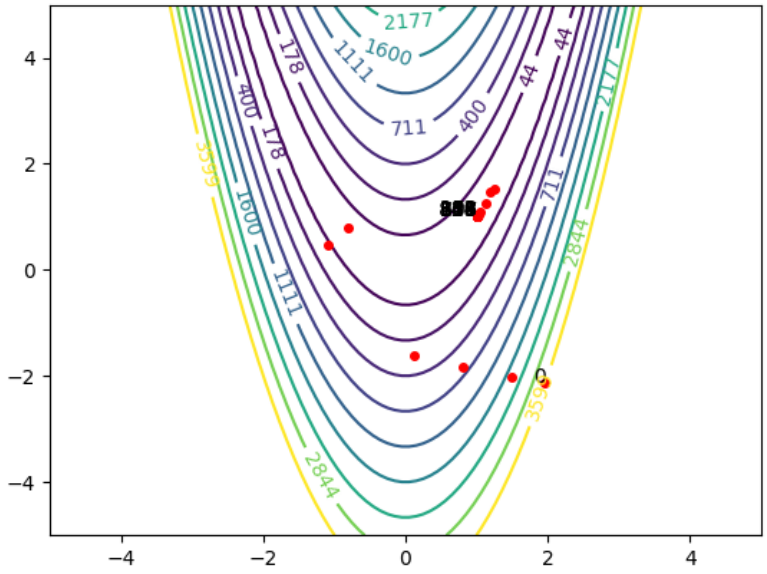
Gradient with momentum III

Back to Rosenbrock, $D = 2$, $x^\star = (1, 1)$, $f(x^\star) = 0$, no linesearch, $\bar{\alpha} = 0.0001$, budget=3000, plots of the best x so far:

gradient



momentum



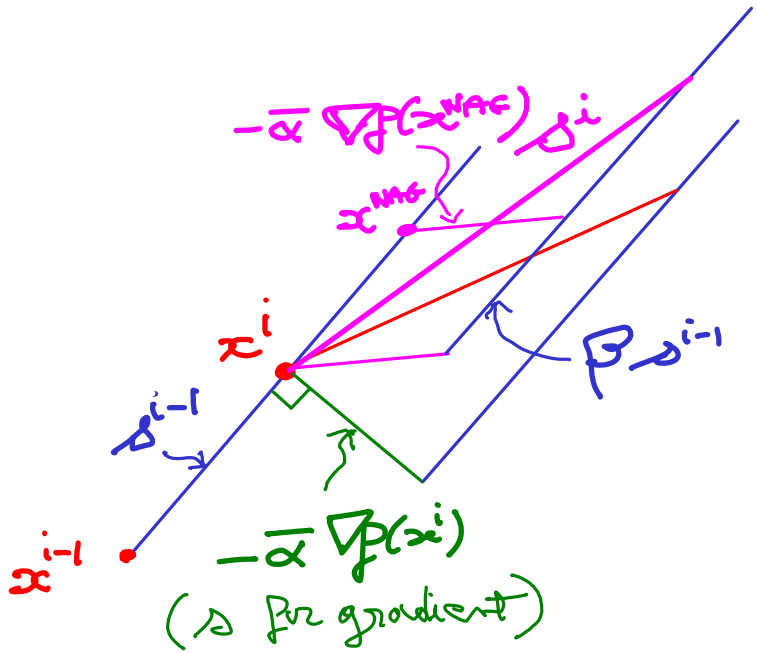
The momentum acceleration allows to find the solution.

Nesterov accelerated gradient (NAG)

Same idea as the momentum direction, but anticipate the position of the next point in the gradient calculation [Nesterov, 1983]:

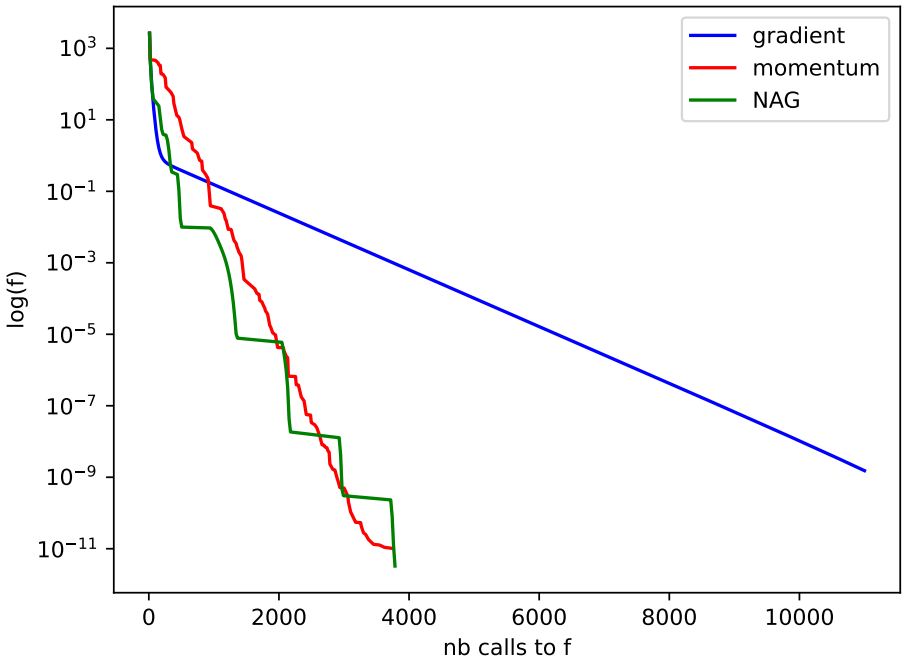
$$s^i = -\bar{\alpha}\nabla f(x^i + \beta(x^i - x^{i-1})) + \beta s^{i-1}$$

On the sketch, ∇f turns upwards and the step is adjusted accordingly.
s^i is no longer necessarily a descent direction.



Comparison of methods (1)

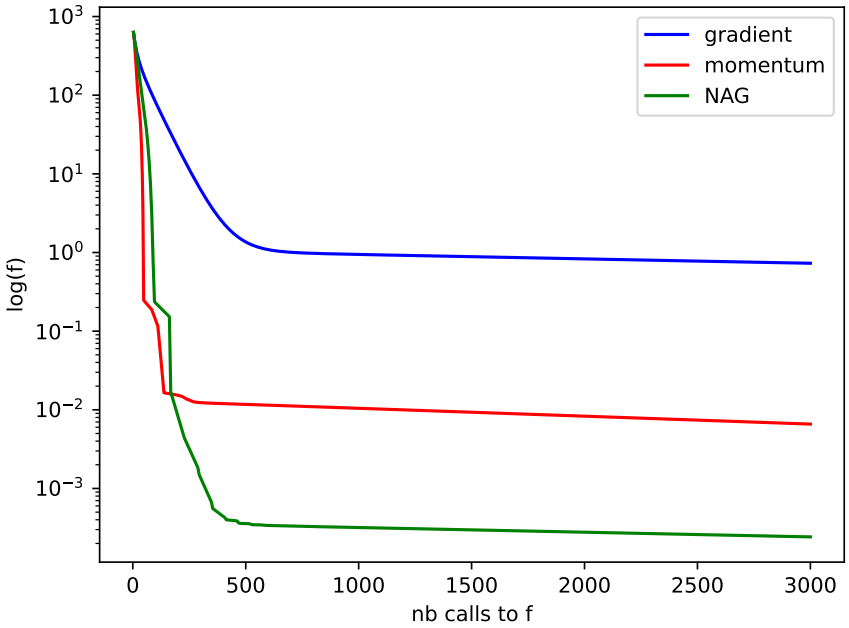
Test convergence speed on quadratic function, $D = 10$, cond. nb. $= 100$, $\bar{\alpha} = 0.01$, no linesearch



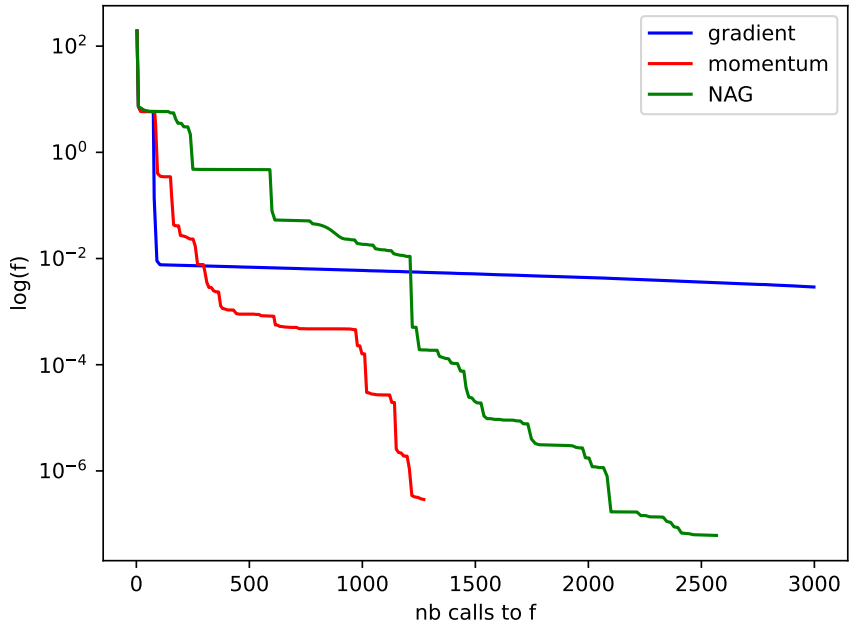
In accordance with theory, convergence speed of NAG is better than momentum which is better than gradient.
Note the plateaus (or "ripples") typical of NAG.
NAG not always better, cf. lab session.

Comparison of methods (2)

Rosenbrock, $D = 2$: ability to handle curved ravines. $\bar{\alpha} = 0.0001$



no linesearch, relative perf depends on starting point but gradient more often fails

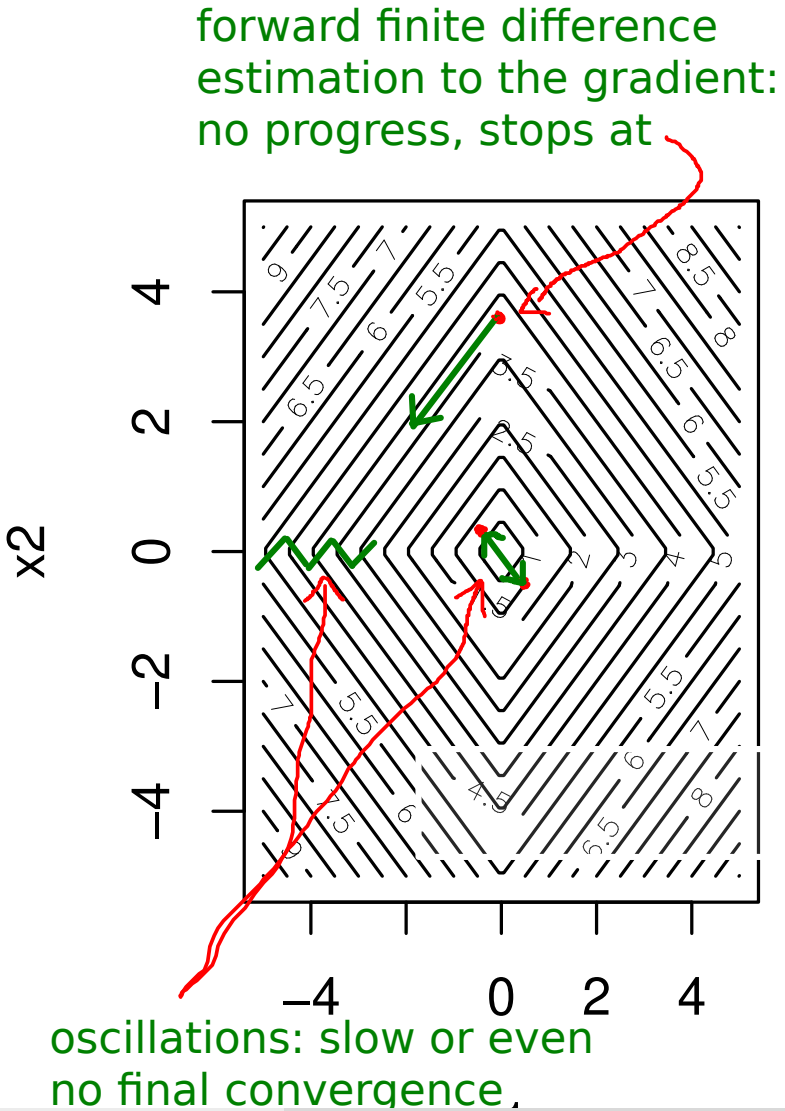


with linesearch, NAG and momentum consistently better than gradient

Nondifferentiable functions and gradient-based descent algorithms

Use on nondifferentiable functions: often converges at points which are not a minimum even on convex functions (e.g., if an iterate is at a kink).

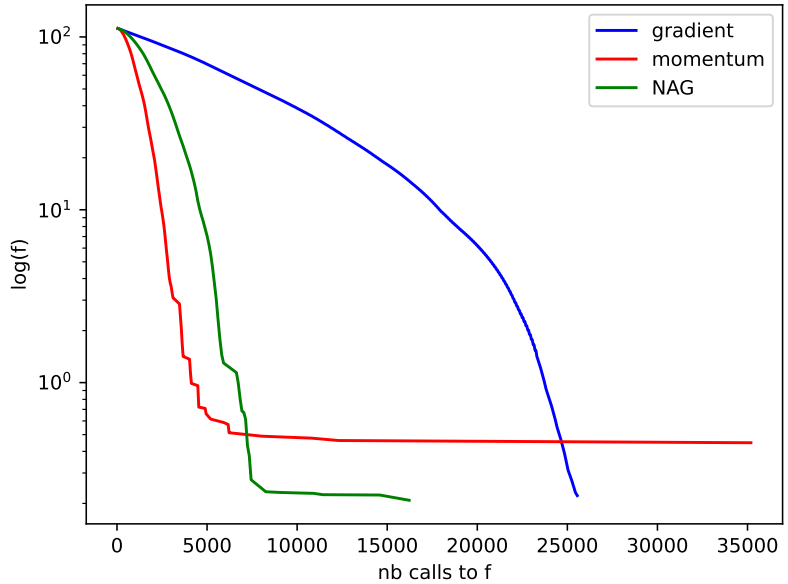
Try function $f(x) = \sum_{i=1}^{D} |x_i|$ ("L1norm") with the code.



Nondifferentiable function : comparison of methods

L1 norm in $D = 50$, $\bar{\alpha} = 0.01$, no linesearch

NAG helps for such function. Yet no convergence is accurate (linesearch is necessary on this function).



Restarted local searches

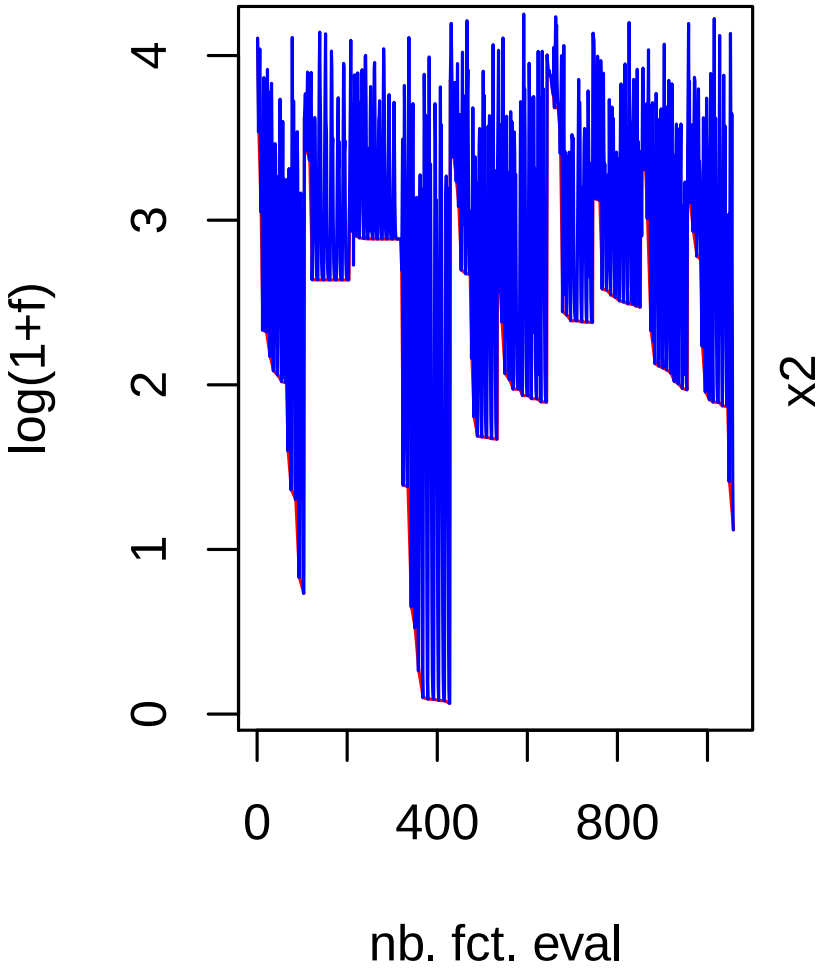
Gradient based descent gets trapped at local optima. Increase robustness by restarting from random initial points.
A mix between 2 extremes: local vs global, line search vs volume search, specific (to unimodal differentiable functions) vs without assumption, efficient vs very slow.
Simple implementation where the max bubget is equally divided between local searches:

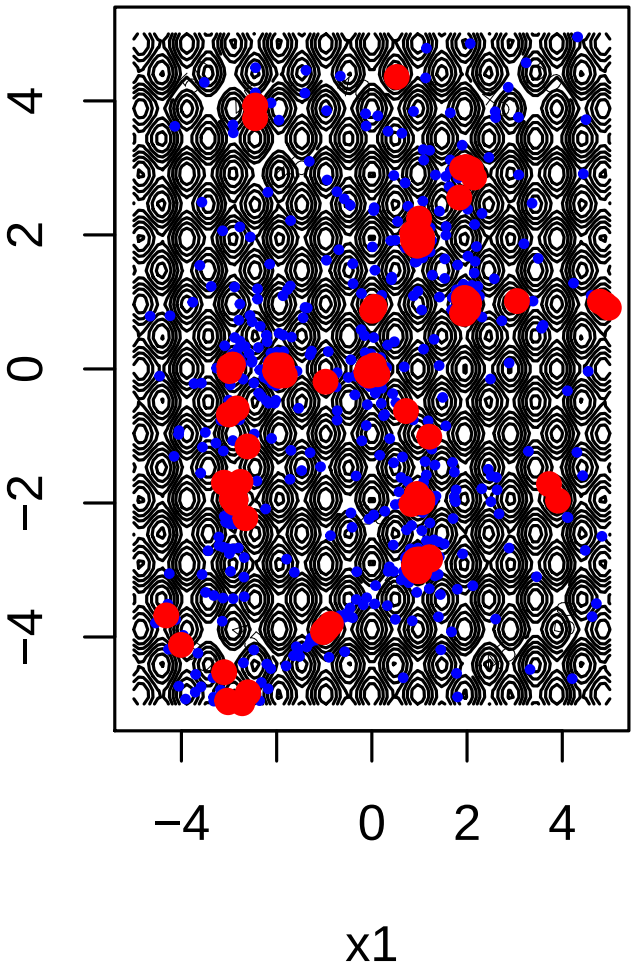
Require: `budget`, `nb_restarts`
  for `i in 1` **to** `nb_restarts` **do**
    `xinit <- runif(n=d,min=LB,max=UB)`
    `res<-gradient_descent(xinit,budget=budget/nb_restarts)`
    update global search results
  end for

Restarted local searches: example

Execution of the `restarted_descent` file.

`fun <-rastrigin, d<-2, budget<-1000, nb_restart<-10:`





Towards ML: regularized quadratic function I

Let's consider a function with a relationship to ML:

$$f(x) = \sum_{i=1}^{D}(x_i - c_i)^2 + \lambda \sum_{i=1}^{D}|x_i| \quad , \quad \lambda \geq 0 \tag{4}$$

First term: sphere function centered at c, $c_i = i$, $i = 1, \ldots, d$.
A simplistic model to the mean square error of a NN (a D-dimensional linear NN with 1 data point):

$$\begin{array}{ccc} 1 \xrightarrow{x_1} x_1 & \approx & c_1 \\ \vdots & & \vdots \\ 1 \xrightarrow{x_D} x_D & \approx & c_D \end{array}$$

Towards ML: regularized quadratic function II

Second term: L1 norm times λ, a.k.a. LASSO regularization. The x_i's would be the weights of a NN. This term helps in improving the test error. Let's see why.
(The relation between λ and the solution to Problem (4) will be studied during the practicum).
Problem (4) can be better understood as a constrained optimization problem.

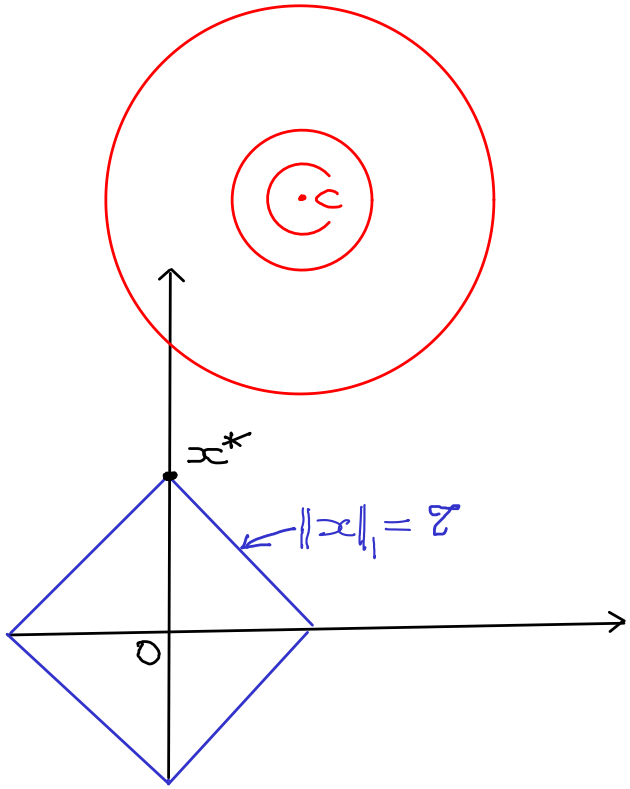
$$\begin{cases} \min_x f(x) = \|x - c\|^2 \\ \text{tel que} \quad g(x) = \|x\|_1 - \tau \leq 0 \quad , \quad \tau > 0 \end{cases}$$

has the following Lagrangian, to be minimized on x

$$\min_x f(x) + \lambda^\star g(x) = \|x - c\|^2 + \lambda^\star \|x\|_1 - \lambda^\star \tau$$

Towards ML: regularized quadratic function III

The first 2 terms are the regularized function (4), the last term doesn't depend on x. Let's draw the sphere function and the constraint boundary $\|x\|_1$:



Towards ML: regularized quadratic function IV

It is seen that the solution tends to be at a vertex of the feasible domain, where some of the x components cancel out. This phenomenon occurs more often when D is large and is less obvious for $D = 2$.
If the x's were a NN weights, setting x_i to 0 is equivalent to cutting the corresponding link, thus reducing the NN capacity, thus making it less prone to overfitting. Finding the best value for λ is something of an art (hyper-parameter tuning, the art of cross-validation).

Partial conclusions on deterministic optimizers

- Numerical optimization is a fundamental technique for quantitative decision making, statistical modeling, machine learning, ...
- The enthousiasm for machine learning has led to very many optimization algorithms which we will not discuss in this course : see for example [Sun et al., 2019, Sra et al., 2012].
- Part 2 of the class: stochastic gradient, a continuation of what was covered which is important for conceptual (introduce stochasticity) and practical (large number of data) reasons.

References I


Bishop, C. M. (2006).
Pattern recognition and machine learning.


Cauchy, A. et al. (1847).
Méthode générale pour la résolution des systèmes d'équations simultanées.
Comp. Rend. Sci. Paris, 25(1847):536–538.


Fukushima, Y., Cayol, V., Durand, P., and Massonnet, D. (2010).
Evolution of magma conduits during the 1998–2000 eruptions of piton de la fournaise volcano, réunion island.
Journal of Geophysical Research: Solid Earth, 115(B10).


Li, M. (2019).
Gradient descent and convergence rate.
`https://youtu.be/LJ05Ujfrsk0`.
video of Berkeley stat 157 course.


Minoux, M. (2008).
Programmation mathématique. Théorie et algorithmes.
Lavoisier.

References II

Nesterov, Y. (1983).
A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$.
In *Doklady an USSR*, volume 269, pages 543–547.

Polyak, B. T. (1964).
Some methods of speeding up the convergence of iteration methods.
USSR computational mathematics and mathematical physics, 4(5):1–17.

Ravikumar, P. and Singh, A. (2017).
Convex optimization.
http://www.cs.cmu.edu/~pradeepr/convexopt/.

Schmidt, M., Fung, G., and Rosales, R. (2007).
Fast optimization methods for l1 regularization: A comparative study and two new approaches.
In *European Conference on Machine Learning*, pages 286–297. Springer.

Sgueglia, A., Schmollgruber, P., Bartoli, N., Atinault, O., Benard, E., and Morlier, J. (2018).
Exploration and sizing of a large passenger aircraft with distributed ducted electric fans.
In *2018 AIAA Aerospace Sciences Meeting*, page 1745.

References III


Sra, S., Nowozin, S., and Wright, S. J. (2012).
Optimization for machine learning.
MIT Press.


Sun, R. (2019).
Optimization for deep learning: theory and algorithms.
arXiv preprint arXiv:1912.08957.


Sun, S., Cao, Z., Zhu, H., and Zhao, J. (2019).
A survey of optimization methods from a machine learning perspective.
IEEE transactions on cybernetics, 50(8):3668–3681.


Vandenberghe, L. (2022).
Optimization methods for large-scale systems.
UCLA course ECE236C, `https://www.seas.ucla.edu/~vandenbe/ee236c.html`.

Proof that Lipschitz gradient and convexity $\Rightarrow$ L-strongly smooth function

Lipschitz gradient : $\|\nabla f(y) - \nabla f(x)\| \leq L\|y - x\|$.
Introduce $x_t = x + t(y - x)$, $t \in [0, 1]$,
$f(y) - f(x) = \int_0^1 \nabla f(x_t) dx_t = \int_0^1 \nabla f(x_t)(y - x)dt$.

$$|f(y) - [f(x) + \nabla f(x)(y - x)]| = |\int_0^1 (\nabla f(x_t) - \nabla f(x))(y - x)dt|$$

$$\text{(Cauchy-Schwartz)} \quad \leq \int_0^1 \|\nabla f(x_t) - \nabla f(x)\|\|y - x\|dt$$

$$\text{(Lipschitz grad)} \quad \leq \int_0^1 L\|x_t - x\|\|y - x\|dt$$

$$= \int_0^1 Lt\|y - x\|^2 dt = \frac{L}{2}\|y - x\|^2$$

By convexity, the interior of the absolute value on the left is positive, therefore $f(y) \leq f(x) + \nabla f(x)(y - x) + \frac{L}{2}\|y - x\|^2$ which is the definition of an L-strongly smooth function. $\square$

Course outline

Optimization for machine learning

A word about constraints

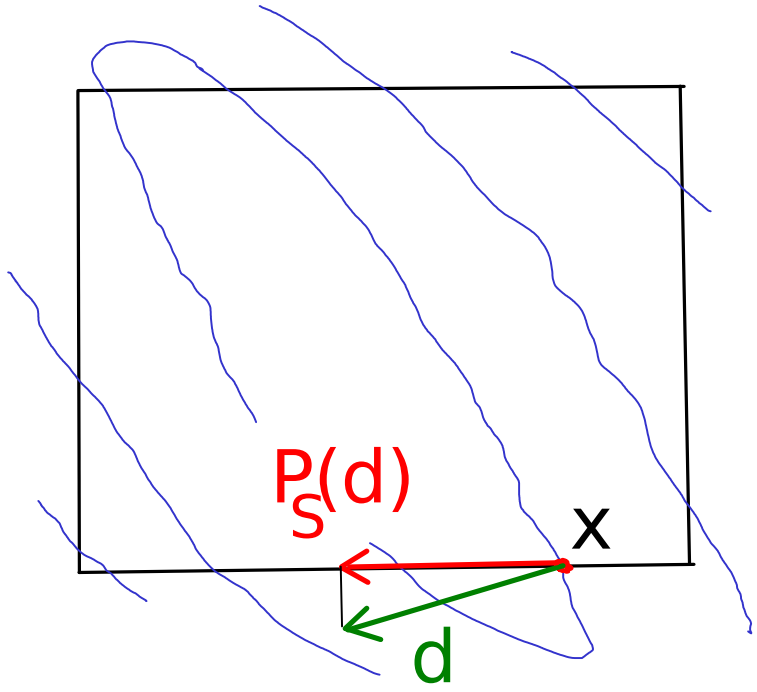
$$\begin{cases} \min_{x \in \mathcal{S}} f(x) \quad , \quad \mathcal{S} = \mathbb{R}^D \\ \text{such that } g_i(x) \leq 0 \quad , \quad i = 1, m \end{cases}$$

Bound constraints

$\mathcal{S}$ is an hypercube of $\mathbb{R}^D$, $\mathcal{S} = [LB, UB] \subset \mathbb{R}^D$. It could be described by constraints, $g_{2i-1}(x) := LB_i - x_i \leq 0$, $g_{2i}(x) := x_i - UB_i \leq 0$, $i = 1, \ldots, D$ but these constraints are so simple that they can be directly handled by projection.

If x^i is at a bound and the search direction d^i takes it outside[a] $\mathcal{S} = [LB, UB]$, project the search direction vector onto the active bound.

QUIZZ Projection on bounds: how to code this projection on the bounds for steps leaving the feasible domain?



[a] This can even happen for a convex function in a convex $\mathcal{S}$, as the drawing shows.

Constraints handling by penalizations (1)

$$\begin{cases} \min_{x \in \mathcal{S} \in \mathbb{R}^d} f(x) \\ \text{such that } g(x) \leq 0 \end{cases}$$

(vector notation for the constraints)

We give two techniques to aggregate f and the g_i's into a new objective function (to minimize).

External penalty function: penalize points that do not satisfy the constraints

$$f_r(x) = f(x) + r \left[\max(0, g(x))\right]^2 \quad , \quad r > 0$$

- Pros: simple, $\nabla f_r()$ continuous accross the constraint boundary (if f and g are) 

Continuous penalty function: prove that $\nabla f_r()$ is continuous accross $g() = 0$ if $\nabla f()$ and $\nabla g()$ are.
- Cons: Convergence by the infeasible domain (hence external), need to find r large enough to reduce infeasibility, but not too large because of numerical issue (high curvature accross constraint)

Constraints handling by penalizations (2)

Lagrangian: for problems without duality gap[2], e.g., convex problems, there exists Lagrange multipliers $\lambda^\star$ such that

$$x^\star \in \arg\min_{x \in \mathcal{S}} L(x; \lambda^\star)$$
$$\text{where } L(x; \lambda^\star) := f(x) + \lambda^\star g(x)$$

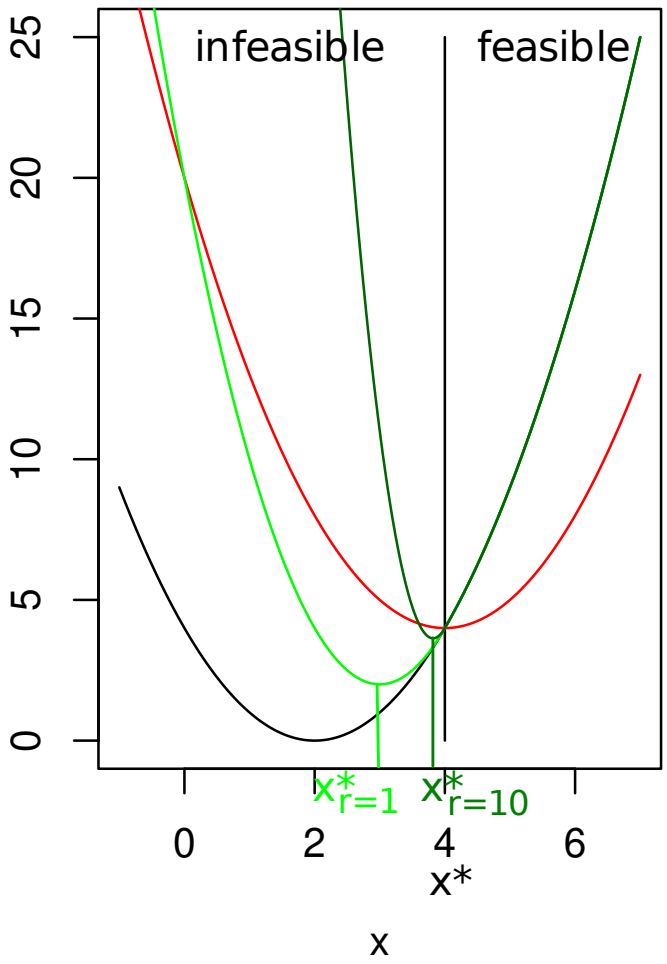
The Lagrangian $L(; \lambda^\star)$ is (when no duality gap) a valid penalty function.

- Pros: duality provides a way to calculate $\lambda^\star$, yields a feasible solution.
- Cons: estimating $\lambda^\star$ has a numerical cost. For most problems with local optima there is a duality gap $\Rightarrow$ rely on augmented Lagrangians[4].

[2] cf. duality, out of scope for this course

Constraints handling by penalizations (3)

Example: $f(x) = (x-2)^2$, $g(x) = 4 - x \leq 0$, $x^* = 4$, convex problem



f and g in black, $L(x; \lambda^\star = 4)$ in red, exterior penalty $f_r()$ with $r = 1$ and 10 in light and dark green, respectively.

The Lagrangian is a valid penalty here.

As r grows, $x_r^\star \to x^\star$ but the curvature of $f_r()$ increases.