① Continous optimization : an Introduction

References : $\begin{cases} \text{Convex Optimization (Stephen Boyd and Lieven Vandenberghe)} \\ \text{Numerical optimization (Nocedal and Wright)} \end{cases}$

* Mathematical problem

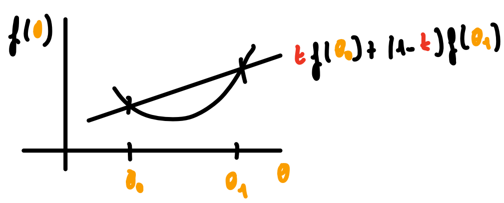
$$\begin{cases} \min_{\theta} & f_0(\theta) \\ \text{s.t.} & f_i(\theta) \leq 0 \\ & i = 1 \dots m \end{cases} \quad \overset{\text{Similarly}}{\Leftrightarrow} \quad \begin{cases} \min_{\theta \in C} & f_0(\theta) \\ \text{where } & C = \{ \theta \mid f_i(\theta) \leq 0 \} \end{cases}$$

Names
- * Optimization
- * Mathematical programming

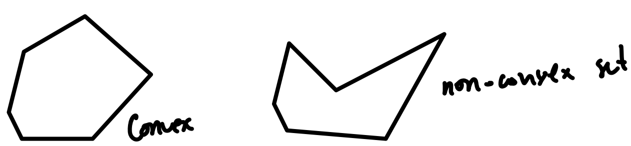
* Convex optimization

* We ask $\begin{cases} f_0(\theta) \text{ to be convex} \\ C \text{ to be a convex set} \end{cases}$

* Convex function : $f(t\theta_0 + (1-t)\theta_1) \leq t f(\theta_0) + (1-t) f(\theta_1)$ $\quad \forall \theta_0, \theta_1$, $\forall t \in [0,1]$



* Convex set : $t\theta_0 + (1-t)\theta_1 \in C$ $\quad \forall \theta_1, \theta_2, \forall t \in [0,1]$



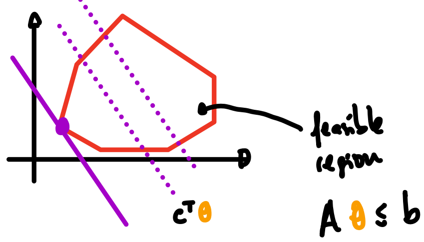
* Examples $\begin{cases} \text{Linear constraints :} & A\theta \leq b \\ \text{Euclidian norms :} & \|\theta\|^2 \leq r \\ \text{Bounds :} & \theta_{lb} \leq \theta \leq \theta_{ub} \end{cases}$

* Not trivial to always identify convex sets
  - * Some operators preserve convexity
  - * There exists tricks to easily identify convex sets

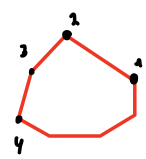
* Why convexity important?
  - If the function is convex, then a local minimum is a global minimum
  - Sometimes we only have local convexity ⇒ global optimum not guaranteed

* <u>First example</u>: Linear programming (convex problem)

$$\begin{cases} \min\limits_{\theta} & c^T\theta \\ \text{s.t.} & A\theta \leq b \end{cases}$$



* lots of applications
* Most of the times θ is really large
* <u>Algorithms</u>
  - * <u>Simplex</u>: optimal solution traversing the edges
  - * <u>Interior point</u>: optimal solution



<u>Unconstrained optimization</u>

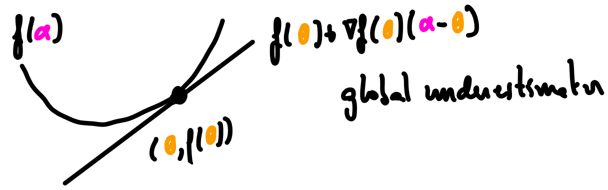
$\min f(\theta)$

- f might be convex
- $f \in C^1$, $\exists\, \nabla f(\theta)\ \forall\theta \to |\theta|$ is not differentiable
- In some applications, constraints are added as a penalization in the cost function
- lots of examples in ML

* First order optimality condition

If f is convex and C^1, then

$$\forall \theta, \alpha \quad f(\theta) + \nabla f^T(\theta)(\alpha - \theta) \le f(\alpha)$$



* Proof in 1D : $f((1-t)\theta + t\alpha) \le (1-t)f(\theta) + t f(\alpha)$

$$\frac{f((1-t)\theta + t\alpha) - f(\theta)}{t} + f(\theta) \le f(\alpha) \overset{\text{taking } t \to 0}{\Rightarrow} f(\theta) + \nabla f^T(\theta)(\alpha - \theta) \le f(\alpha)$$

* Taylor approximation of $f(\alpha)$ at θ

* If the function is convex,
local information like function and gradient
at a point θ provides global information (global underestimate)

* If $\exists\, \theta$ s.t. $\nabla f(\theta) = 0 \Rightarrow f(\theta) \le f(\alpha) \quad \forall \alpha \Rightarrow \theta$ global minimizer of f

* If f is not convex, but locally convex and C^2, a taylor expansion gives

$$f(\alpha) = f(\theta) + \nabla f^T(\theta)(\alpha - \theta) + \frac{1}{2}(\alpha - \theta)^T \nabla^2 f(\theta)(\alpha - \theta) + \dots .$$

If $\exists\, \theta^*$ s.t. $\nabla f(\theta^*) = 0$, then for $\forall \alpha$ s.t. $(\alpha - \theta^*)^T \nabla^2 f(\theta)(\alpha - \theta^*) \ge 0$

we have $f(\theta^*) = f(\alpha) - (\alpha - \theta^*)^T \nabla^2 f(\theta)(\alpha - \theta^*) \le f(\alpha)$

* Second example : least squares problem

$$\min_{\theta} \; f_0(\theta) = \| A\theta - b \|_2^2 = (A\theta - b)^T (A\theta - b) = (A_{ij}\theta_j - b_i)(A_{ik}\theta_k - b_i)$$

Gateaux derivative :

$$\frac{\partial f_0}{\partial \theta_m}(\tilde{\theta}) = A_{ij}\delta_{jm}\tilde{\theta}(A_{ik}\theta_k - b_i) + (A_{ij}\theta_j - b_i)(A_{ik}\delta_{km}\tilde{\theta})$$

$$= \left[A_{im}[A_{ik}\theta_k - b_i] + A_{im}[A_{ik}\theta_k - b_i]\right] = 2\left[A^T(A\theta - b)\right]\tilde{\theta} = 0$$

We have to solve $[A^TA]\theta = A^Tb$ (Normal equations)

* A^TA is always square and symmetric.
* Case A square $\Rightarrow$ same as $A\theta = b$
* Case A has vertical shape (overdetermined) $A = [\;]$
  ↳ Minimize the error in the columns of A
* Case A has horizontal shape (underdetermined) $A = [\quad\quad]$
  ↳ A^TA has not full rank $\Rightarrow$ Ill-posed problem
  ↳ Plenty of solution
  ↳ In ML terms is an example of overfitting

* Algorithms
  * Gradient method
    * $\theta_{k+1} = \theta_k - \tau \nabla f(\theta_k)$ where $\tau > 0$ is small enough
    * τ is called line-search (or learning rate in ML)
    * Descend direction: $f(\theta_{k+1}) = f(\theta_k) + \nabla f(\theta_k)^T(\theta_{k+1} - \theta_k) + \dots$

      $= f(\theta_k) - \tau \|\nabla f(\theta_k)\|^2 \leq f(\theta_k)$

    * Steepest descend (Euclidean norm): find $\Delta\theta$ s.t.

      $$\begin{cases} \min \nabla f(\theta_k)^T \Delta\theta \\ \text{s.t. } |\Delta\theta| \leq 1 \end{cases} \Rightarrow |\Delta\theta| = \frac{-\nabla f(\theta_k)}{\|\nabla f(\theta_k)\|}$$

    * Stopping criteria: $\|\nabla f(\theta_k)\| \leq \eta$

* <u>Line-search</u> (learning-rate)

* Exact line-search

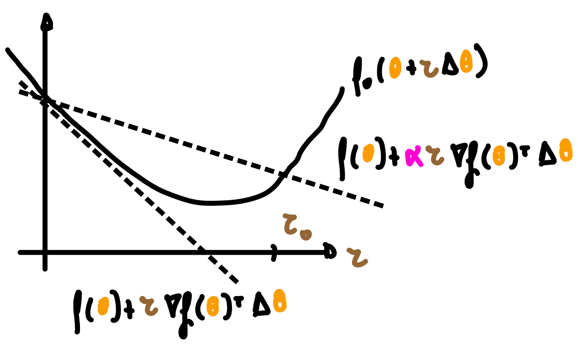
Find τ along the ray $\{\theta + \tau\Delta\theta \,;\, \tau \geq 0\}$:

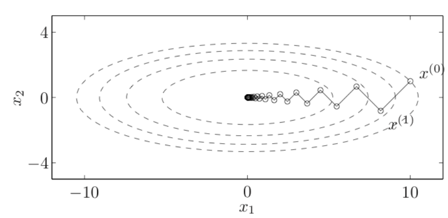
$$\tau = \arg\min_{\tau \geq 0} f_0(\theta + \tau\Delta\theta)$$

- Usually used when computing the cost is much cheaper than computing the gradient or direction
- Sometimes can be computed analitically or very efficiently

* Backtracking line-search (Inexact)
  * Approximately minimize f_0 or reduce f_0 "enough"
  * Given $\Delta\theta$, take $\tau = \tau^*$; $\alpha = (0, 0.5)$, $\beta \in (0,1)$

$$\text{while } f_0(\theta_k + \tau\Delta\theta_k) > f_0(\theta_k) + \alpha\tau\nabla f(\theta_k)^T \underbrace{\Delta\theta_k}_{-\tau\nabla f(\theta_k)} ;$$
$$\tau = \beta\tau$$





For small τ:

$$f_0(\theta_k + \tau\Delta\theta_k) \approx f_0(\theta_k) + \tau \overbrace{\nabla f_0^T(\theta_k)\underbrace{\Delta\theta_k}_{-\tau\nabla f(\theta_k)}}^{<0} < f_0(\theta_k) + \tau\alpha\nabla f_0^T(\theta_k)\Delta\theta_k$$

α: Fraction of the decrease in f predicted by linear extrapolation that will be accept. Typically between $\alpha \in [0.01, 0.3]$

β: Between $\beta \in [0.1, 0.8]$. Typicall number $\beta = 0.5$

* Steepest descent for quadratic norms

Quadratic norm: $\|\theta\|_P = (\theta^T P \theta)^{1/2} = \|P^{1/2}\theta\|_2$

- $\left\{ \begin{array}{l} \min \ \nabla f_0(\theta_k)^T \Delta\theta \\ \text{s.t. } \|\Delta\theta\|_P \le 1 \end{array} \right. \Rightarrow \Delta\theta = -P^{-1}\nabla f_0(\theta_k)$

$\theta_{k+1} = \theta_k - \tau P^{-1} \nabla f_0(\theta_k)$

- If f_0 is least squares of linear system of equation: P is called preconditioner
- Interesting when P^{-1} is cheap
- Change of coordinates to better condition the problem

* Newton's method

- $\theta_{k+1} = \theta_k - [\nabla^2 f_0(\theta_k)]^{-1} \nabla f(\theta_k)$

- Descend direction:

$$f(\theta_{k+1}) = f(\theta_k) + \nabla f^T(\theta_k)\Delta\theta_k + \dots \quad \text{if } \nabla^2 f_0(\theta_k) \text{ s.p.d.}$$

$$= f(\theta_k) - \nabla f^T(\theta_k)[\nabla^2 f_0(\theta_k)]\nabla f(\theta_k) \le f(\theta_k)$$

- Minimization of second-order approximation

$$\left\{ \begin{array}{l} f(\theta_{k+1}) \simeq f(\theta_k) + \nabla f^T(\theta_k)\Delta\theta_k + \Delta\theta_k^T[\nabla^2 f_0(\theta_k)]\Delta\theta_k \\ \min f(\theta_{k+1}) \simeq \min\limits_{\Delta\theta_k} f(\theta_k) + \nabla f^T(\theta_k)\Delta\theta_k + \Delta\theta_k^T[\nabla^2 f_0(\theta_k)]\Delta\theta_k \\ \quad \hookrightarrow \Delta\theta_k = -[\nabla^2 f_0(\theta_k)]^{-1}\nabla f(\theta_k) \end{array} \right.$$

* Steepest descend in Hessian norm

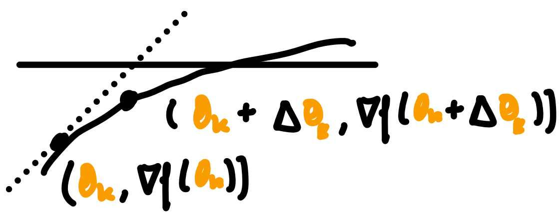
$$\begin{cases} \min \ \nabla f_0(\theta_k)^T \Delta\theta \\ \text{s.t. } \|\Delta\theta\|_{\nabla^2 f} \leq 1 \end{cases} \quad \Delta\theta = -\underbrace{\left[\nabla^2 f_0(\theta_k)\right]^{-1}}_{P} \nabla f_0(\theta_k)$$

it can be seen as preconditioner

* Solution of linearized optimality condition

$$\nabla f(\theta_k + \Delta\theta_k) \simeq \nabla f(\theta_k) + \nabla^2 f(\theta_k)\,\Delta\theta_k = 0 \Rightarrow \Delta\theta = -\left[\nabla^2 f_0(\theta_k)\right]^{-1} \nabla f_0(\theta_k)$$

In 1D:



* { Highly used in computational engineering
  Rarely used in ML

* { Convergence when closed to the solution
  Quadratic convergence (at most 6/7 iterates)
  Stopping criteria: $\lambda(\theta_k) = \nabla f^T(\theta_k) \underbrace{\left[\nabla^2 f_0(\theta_k)\right]^{-1} \nabla f(\theta_k)}_{\Delta\theta_k} \leq \varepsilon$

* Implicit method
  - Descend method has some similarities with ODEs
  - Gradient is explicit
  - Newton and quadratic are implicits
  - { Explicits { Each iter fast
                  many iterations
      Implicits { Each iter expensive (in general)
                  Few iteration

* Quasi-newton
  { · Approximation of the hessian with gradients