Méthodes de descente : l'algorithme du gradient

Par Jérôme MONNIER, Professeur à l'INSA de Toulouse,
Département de Génie Mathématique et Modélisation.

Plan :

- Bref rappel en images de la convexité, minimun local / global,
- Lien minimisation - système linéaire (cas quadratique),
- Les algorithmes de descente,
- L'algorithme du gradient à pas optimal.

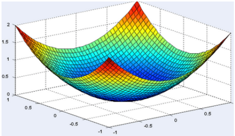
Optimisation : fonctionnelles convexes ou non, minimum local / global

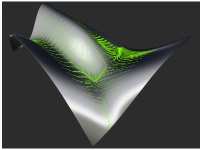
Objectif :

$$\boxed{\left\{ \begin{array}{c} \text{Trouver } x^* \in \mathbb{R}^n \text{ tel que :} \\ j(x^*) = \min_{x \in \mathbb{R}^n} j(x) \end{array} \right.}$$

avec $j : \mathbb{R}^n \to \mathbb{R}$, j de classe C^1 au moins (C^2 si besoin).

Illustration cas $n = 2$: une fonction quadratique strictement convexe (le "cas idéal"), et une fonctionnelle avec minima locaux...





Condition Nécéssaire : $\boxed{\nabla j(x) = 0}$

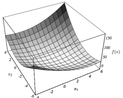
Cas j strictement convexe : cette condition devient Suffisante.

Lien minimisation - système linéaire

$\exists$ lien entre $Ax = b$, A sym. déf. positive, et $\min\limits_{x} j(x)$
$\rightarrow$ *j fonctionnelle quadratique associée* (j : fctelle d'*énergie*).

> *Proposition*. Soit la fonctionelle j définie de $\mathbb{R}^n$ dans $\mathbb{R}$ par :
> $j(x) = \frac{1}{2}(Ax, x) - (b, x)$
> avec A matrice symétrique définie positive.
> Alors il existe une unique solution $x^* \in \mathbb{R}^n$, et cette solution vérifie : $Ax = b$.

Preuve. $j(x) = \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n a_{ij}x_ix_j - \sum_{i=1}^n b_ix_i$.
$\rightarrow$ *j est quadratique donc strictt cvx.*
$\Rightarrow$ *unique minimum global dans $\mathbb{R}^n$.*
C.N. (et suffisante) de min. : $\nabla j(x) = 0$.
Or : $\forall i \in \{1, .., n\}, \partial_i j(x) = \sum_{j=1}^n a_{ij}x_j - b_ix_i = (Ax)_i$, *d'où le résultat*$\square$
NB. Hessienne : $D^2 j(x) = A, \ \forall x \in \mathbb{R}^n$.



Algorithmes de descente : principe

x_0 donné (le mieux choisi possible...), on définit x_{n+1} tq :

$$j(x^{(n+1)}) < j(x^{(n)}) \text{ (on descend...)}$$

Pour cela, on doit déterminer :

- la *direction de descente* $d^{(n)}$,
- le *pas de descente* $\alpha^{(n)}$ (jusqu'où descend-on ?)

Puis : $\boxed{x^{(n+1)} = x^{(n)} + \alpha^{(n)} d^{(n)} \ , \quad n \geq 0}$

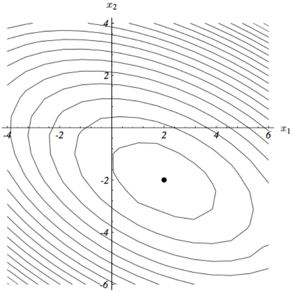


Figure 3: Contours of the quadratic form. Each ellipsoidal curve has constant $f(x)$.

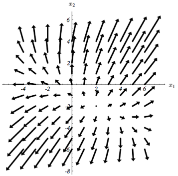


Figure 4: Gradient $f'(x)$ of the quadratic form. For every x, the gradient points in the direction of steepest increase of $f(x)$, and is orthogonal to the contour lines.

Algorithmes de descente : principe (2)

- Pas de descente α : *recherche linéaire*.
- Direction de descente d : basée sur l'information du gradient $\nabla j(x)$, et tq : $\boxed{<\nabla j(x), d> < \ 0}$

En effet,
$$
\begin{aligned}
j(x^{(n+1)}) = j(x^{(n)}) + &< \nabla j(x^{(n)}), x^{(n+1)} - x^{(n)} > \dots \\
&+ < D^2 j(x^{(n)} \cdot (x^{(n+1)} - x^{(n)}), (x^{(n+1)} - x^{(n)}) > \dots \\
&+ \mathcal{O}(\|x^{(n+1)} - x^{(n)}\|^3)
\end{aligned}
$$
soit :

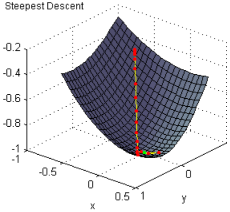
$$\boxed{j(x^{(n+1)}) - j(x^{(n)}) = \alpha^{(n)}) < \nabla j(x^{(n)}), d^{(n)}) > + \mathcal{O}(\|x^{(n+1)} - x^{(n)}\|^2)}$$

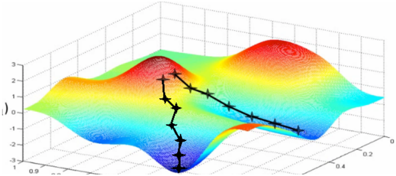
Donc pour α suffisamment petit, $\alpha > 0$, on a bien : $j(x^{(n+1)}) < j(x^{(n)})$.

Algorithmes de descente : convergence et arrêt

Choix de $x^{(0)}$.

- Cas j strictement cvx : facile, ca descend forcément où il faut !
- Cas j admet plusieurs minima locaux : un algo. des descente conduit au minimum local du “bassin versant courant”...





Arrêt algorithme. Deux critères standards :

$$\|\nabla j(x^{(n+1)})\| < \varepsilon_1 \text{ et } \frac{j(x^{(n+1)}) - j(x^{(n)})}{j(x^{(n)})} < \varepsilon_2$$

L'algorithme du gradient à pas optimal

o $\boxed{d^{(n)} = -\nabla j(x^{(n)}) \;, \quad \forall n \geq 0}$ i.e. la plus profonde descente,

o α pas optimal : minimise j *selon la direction* d,

$$\boxed{\alpha^{(n)} \in \mathbb{R}^+ \text{ tel que : } j(x^{(n)} + \alpha^{(n)} d^{(n)}) = \min_{\alpha \in \mathbb{R}^+} j(x^{(n)} + \alpha d^{(n)})}$$

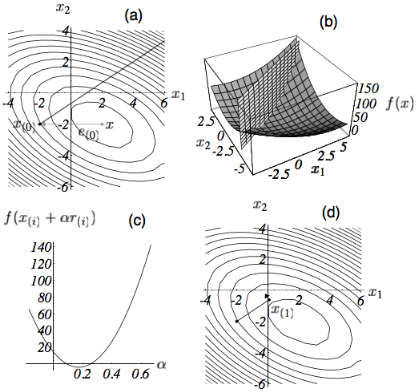


Figure 6: The method of Steepest Descent. (a) Starting at $[-2,-2]^T$, take a step in the direction of steepest descent of f. (b) Find the point on the intersection of these two surfaces that minimizes f. (c) This parabola is the intersection of surfaces. The bottommost point is our target. (d) The gradient at the bottommost point is orthogonal to the gradient of the previous step.

L'algorithme du gradient : convergence

Théorème. Soient A sym. déf. positive et
$j(x) = \frac{1}{2}(Ax, x) - (b, x)$.
Alors, la méth. gradient à pas optimal CV pour tout x_0.
De plus,

$$\|x^{(n+1)} - x^*\|_A \leq \left(\frac{cond_2(A) - 1}{cond_2(A) + 1}\right)^n \|x^{(0)} - x^*\|_A$$

Rappels : $\|x\|_A^2 = (Ax, x)$ et $cond_2(A) = \|A\|_2\|A^{-1}\|_2 = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}$.

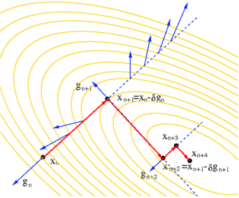
$\rightarrow$ La vitesse de CV dépend de $cond_2(A)$.
$\Rightarrow$ *Préconditionnement* préalable du système potentiellement intéréssant (e.g. l'algo. du *gradient conjugué préconditionné*).

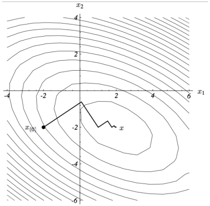
L'algorithme du gradient : convergence (2)

Proposition. A chaque itération, on a :

$$d_n \perp d_{n-1}$$

$\Rightarrow$ Zig-zag !





$\Rightarrow$ Méthode performante de référence :
la méthode du *gradient conjugué* (avec préconditionnement éventuel).