OPTIMISATION ET CONTRÔLE

Grégoire Allaire
Département de mathématiques appliquées
Ecole Polytechnique

Algorithmes d'optimisation sans contrainte (suite et fin)

1. Algorithme de Newton (ordre 2)
2. Autres algorithmes d'ordre 1: vitesse de convergence

I - Algorithme de Newton (ordre 2)

Soit $J(v)$ une fonction de classe C^2 de $\mathbb{R}^N$ dans $\mathbb{R}$.

Idée: si J'' est une matrice définie positive, on fait l'approximation

$$J(w) \approx J(v) + J'(v) \cdot (w - v) + \frac{1}{2} J''(v)(w - v) \cdot (w - v),$$

dont le minimum est $w = v - \left(J''(v)\right)^{-1} J'(v)$.

Algorithme: à partir d'une initialisation $u^0 \in \mathbb{R}^N$ on itère

$$u^{n+1} = u^n - \left(J''(u^n)\right)^{-1} J'(u^n)$$

- Nécessite d'inverser un système linéaire de matrice $J''(u^n)$.
- En fait c'est un algorithme de recherche de zéro de J'...
- Les zéros de J' peuvent être des minima, des maxima ou des points selle...

Algorithme de Newton (2)

On décrit l'algorithme pour la recherche d'un zéro de $F(v) = J'(v)$.

$$u^{n+1} = u^n - \left(F'(u^n)\right)^{-1} F(u^n)$$

Proposition. Soit F une fonction de classe C^2 de $\mathbb{R}^N$ dans $\mathbb{R}^N$, et u un zéro régulier de F (i.e. $F(u) = 0$ et $F'(u)$ inversible). Il existe un réel $\epsilon > 0$ et une constante $0 < C < 1/\epsilon$ tels que, si l'initialisation u^0 vérifie $\|u - u^0\| \leq \epsilon$, alors la méthode de Newton converge

$$\|u^{n+1} - u\| \leq C\|u^n - u\|^2 \quad \text{et} \quad \|u^n - u\| \leq C^{-1}(C\epsilon)^{2^n}.$$

Remarque: ne converge que si u^0 est proche de la solution !
La convergence, dite quadratique, est extrêmement rapide !

Preuve

Par continuité il existe $\delta, C_1, C_2 > 0$ tel que

$$\| \left(F'(v)\right)^{-1} \| \leq C_1, \quad \|F''(v)\| \leq C_2, \quad \text{pour tout } \|v-u\| \leq \delta.$$

Supposons que $\forall n$, $\|u-u^n\| \leq \delta$. Donc $F'(u^n)$ est inversible et

$$u^{n+1} - u = u^n - u - \left(F'(u^n)\right)^{-1} \left(F(u^n) - F(u)\right)$$

qui, par développement de Taylor autour de u^n, devient

$$u^{n+1}-u = \frac{1}{2} \left(F'(u^n)\right)^{-1} \left(\int_0^1 F''(u^n + s(u-u^n))\, ds \right) (u^n-u)\cdot(u^n-u).$$

On majore pour obtenir

$$\|u^{n+1} - u\| \leq C \|u^n - u\|^2 \quad \text{avec} \quad C = \frac{1}{2} C_1 C_2.$$

On choisit $0 < \epsilon < \min(\delta, C^{-1})$ et $\|u-u^0\| \leq \epsilon$. Par récurrence, on obtient bien que

$$\|u-u^n\| \leq \epsilon \leq \delta\,.$$

Preuve (fin)

$$\|u^{n+1} - u\| \leq C\|u^n - u\|^2 .$$

En prenant le logarithme de l'inégalité précédente, on obtient

$$\log \|u^{n+1} - u\| \leq \log C + 2 \log \|u^n - u\|,$$

d'où l'on déduit

$$\log \|u^n - u\| \leq 2^n \log \|u^0 - u\| + (2^n - 1) \log C$$

c'est-à-dire

$$\|u^n - u\| \leq C^{-1} \Big(C\|u^0 - u\| \Big)^{2^n} \leq C^{-1} (C\epsilon)^{2^n}$$

qui converge vers zéro puisque $C\epsilon < 1$.

Algorithme de Newton (variante hybride)

Comme l'algorithme de Newton ne converge **que si l'initialisation est proche de la solution**, il faut hybrider l'algorithme pour qu'il soit robuste et converge toujours.

Une idée possible: on rajoute un pas de descente $0 < \mu \leq 1$.

$$u^{n+1} = u^n - \mu \left(J''(u^n)\right)^{-1} J'(u^n)$$

Stratégie: on démarre avec μ petit, puis on augmente μ jusqu'à la valeur 1.

Lemme. Si J est convexe (et toujours $J''(u)$ inversible), alors $w^n = \left(J''(u^n)\right)^{-1} J'(u^n)$ est une direction de descente pour μ petit.

Preuve. Un développement de Taylor conduit à

$$J(u^{n+1}) = J(u^n) - \mu \left(J''(u^n)\right)^{-1} J'(u^n) \cdot J'(u^n) + \mathcal{O}(\mu^2)$$

avec $\left(J''(u^n)\right)^{-1} J'(u^n) \cdot J'(u^n) > 0$ si $J'(u^n) \neq 0$.

Algorithme de Gauss-Newton

Une variante de Newton pour un problème spécifique.

On considère un problème de moindres carrés non-linéaire

$$\min_{u\in\mathbb{R}^N} \|F(u)\|^2 \quad \text{avec } F(u) = (F_1(u), \cdots, F_M(u))$$

où la fonction $F : \mathbb{R}^N \to \mathbb{R}^M$ est C^2. Si $F(u) = Au - b$, on retrouve les moindres carrés linéaires.
Idée: comme pour Newton, à chaque itération n on approche

$$F(u) \approx F(u^n) + \nabla F(u^n)(u - u^n)$$

et u^{n+1} est un point de minimum de

$$\min_{u\in\mathbb{R}^N} \|A^n u - b^n\|^2$$

$A^n = \nabla F(u^n)$ matrice $M \times N$ et $b^n = \nabla F(u^n)u^n - F(u^n) \in \mathbb{R}^M$.

Algorithme de Gauss-Newton (2)

On suppose que $\ker \nabla F(u^n) = \{0\}$, alors la matrice $(\nabla F(u^n))^* \nabla F(u^n)$ est inversible et u^{n+1} est donné par

$$u^{n+1} = u^n - \Big((\nabla F(u^n))^* \nabla F(u^n) \Big)^{-1} (\nabla F(u^n))^* F(u^n)$$

S'il existe un zéro régulier u de F et si l'initialisation u^0 est proche, on peut démontrer la convergence quadratique de l'algorithme de Gauss-Newton.

Remarque. Si $N = M$ on retrouve exactement l'algorithme de Newton.

Motivation:

1. accélérer la convergence des simples méthodes de gradient,
2. minimiser le coût de calcul des algorithmes,
3. tirer partie de la structure des problèmes d'optimisation.

Idées: utiliser plusieurs pas (u^{n+1} fonction de u^n et u^{n-1}) ou plusieurs directions (mais toujours d'ordre 1).

But du cours: montrer la créativité des algorithmes !
Pas faire un catalogue de recettes...

Problème d'optimisation sans contrainte:

$$\inf_{v \in V} J(v)$$

Vitesse de convergence du gradient à pas fixe

Rappel sur l'algorithme du gradient à pas fixe.
Initialisation: u^0 dans V. Itérations: pour $n \geq 0$

$$u^{n+1} = u^n - \mu\, J'(u^n)\,.$$

Théorème. On suppose que J est α-convexe, différentiable et que J' est L-Lipschitzien sur V, pour $L > 0$

$$\|J'(v) - J'(w)\| \leq L\|v - w\| \qquad \forall v, w \in V.$$

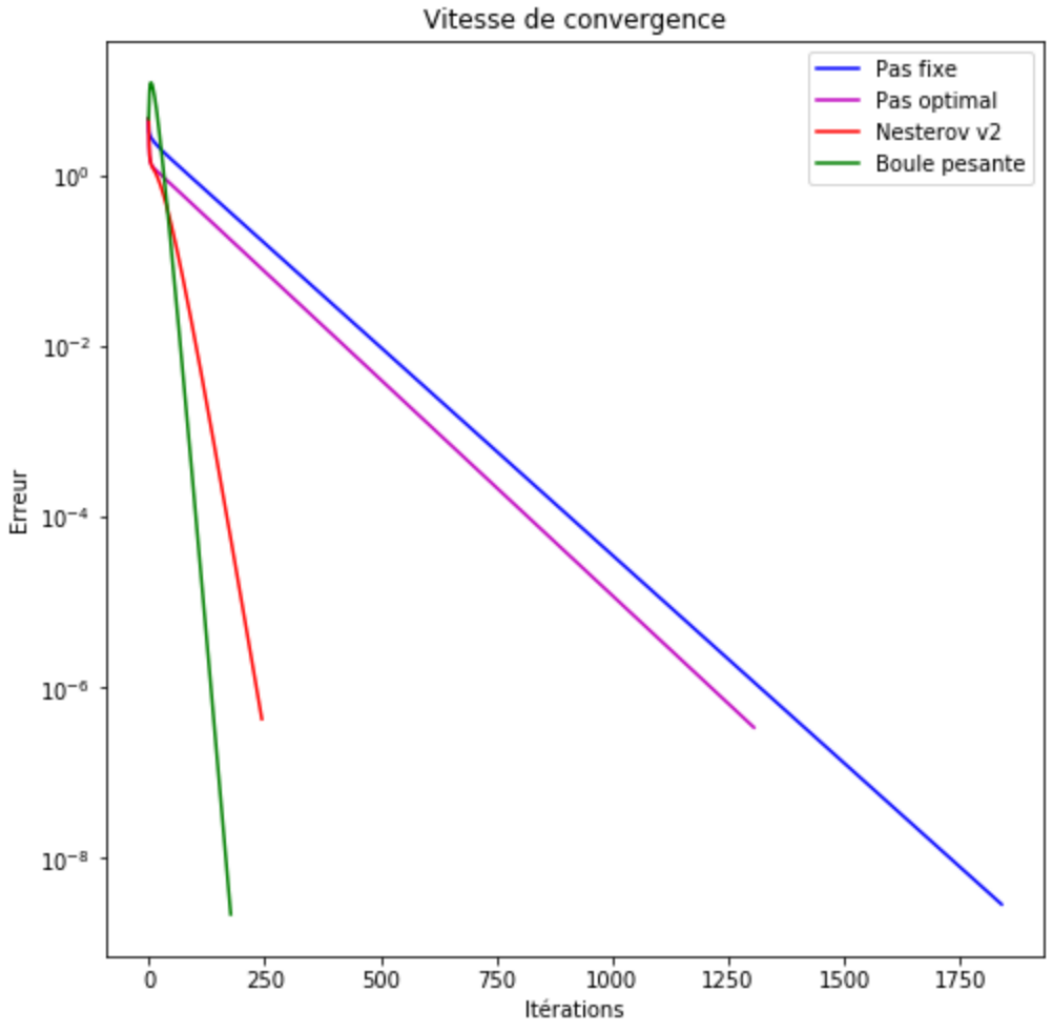
Alors, si $0 < \mu < 2\alpha/L^2$, l'algorithme de gradient à pas fixe converge. En particulier pour $\mu = \alpha/L^2$ on a

$$\|u^n - u\| \leq \gamma^n \|u^0 - u\| \quad \text{avec } \gamma = \sqrt{1 - \frac{\alpha^2}{L^2}} \quad \text{pour } \mu = \frac{\alpha}{L^2}.$$

Est-ce que cette vitesse de convergence est optimale ? **Non !**

Vitesse de convergence des algorithmes d'ordre 1

On dit parfois que la convergence est linéaire car on trouve une droite quand on trace $\log \|u^n - u\|$ en fonction de n.



Interprétation des paramètres α et L

Exemple. Soit A une matrice symétrique, réelle, définie positive de taille $N \times N$ et de valeurs propres $0 < \lambda_1 \leq \cdots \leq \lambda_N$.
Pour $b \in \mathbb{R}^N$, on définit sur $\mathbb{R}^N$

$$J(u) = \frac{1}{2} Au \cdot u - b \cdot u.$$

Alors $\alpha = \lambda_1$ et $L = \lambda_N$.

Lemme. Soit J une fonction de classe C^2 sur V, telle qu'il existe $0 < m \leq M$ et pour tout $u, w \in V$

$$m\|w\|^2 \leq J''(u)(w, w) \leq M\|w\|^2.$$

Alors J est α-convexe avec $\alpha = m$ et J' est L-Lipschitzien avec $L = M$.

Preuve du Lemme

Preuve. Pour $0 \leq t \leq 1$, on définit $j(t) = J(u + t(v-u))$ qui vérifie

$$j'(1) - j'(0) = \int_0^1 j''(t)\, dt\,.$$

Comme

$$j'(t) = \langle J'(u+t(v-u)), v-u\rangle \quad j''(t) = J''(u+t(v-u))(v-u, v-u)$$

on en déduit la m-convexité

$$j'(1) - j'(0) = \langle J'(v) - J'(u), v-u\rangle = \int_0^1 j''(t)\, dt \geq m\|v-u\|^2.$$

Par ailleurs, on définit $k(t) = \langle J'(u + t(v-u)), w\rangle$ qui vérifie

$$k(1) - k(0) = \int_0^1 k'(t)\, dt \quad \text{et} \quad k'(t) = J''(u+t(v-u))(w, v-u).$$

Preuve du Lemme (fin)

On en déduit la M-Lipschitziannité de J'

$$k(1) - k(0) = \langle J'(v) - J'(u), w \rangle = \int_0^1 k'(t)\, dt \leq M\|v-u\|\|w\|,$$

en prenant $w = J'(v) - J'(u)$. En effet, rappelons que

$$J''(u)(w,w) \leq M\|w\|^2 \quad \forall\, w \in V$$

implique

$$|J''(u)(w_1,w_2)| \leq M\|w_1\|\|w_2\| \quad \forall\, w_1, w_2 \in V.$$

Il suffit de prendre $\tilde{w}_i = w_i/\|w_i\|$ et d'écrire

$$4J''(u)(\tilde{w}_1,\tilde{w}_2) = J''(u)\,(\tilde{w}_1+\tilde{w}_2, \tilde{w}_1+\tilde{w}_2) - J''(u)\,(\tilde{w}_1-\tilde{w}_2, \tilde{w}_1-\tilde{w}_2)$$

d'où l'on déduit

$$|J''(u)(\tilde{w}_1,\tilde{w}_2)| \leq \frac{M}{4}\left(\|\tilde{w}_1+\tilde{w}_2\|^2 + \|\tilde{w}_1-\tilde{w}_2\|^2\right) = M.$$

Vitesse de convergence optimale

On se place en dimension finie $V = \mathbb{R}^N$.

Proposition. On suppose que J est de classe C^2, α-convexe et que J' est L-Lipschitzien. Alors l'algorithme de gradient à pas fixe converge pour $0 < \mu < 2/L$, et la vitesse de convergence optimale s'obtient pour le pas $\mu_{\rm opt} = 2/(L+\alpha)$

$$\|u^n - u\| \leq \gamma_{\rm opt}^n \|u^0 - u\| \quad \text{avec} \quad \gamma_{\rm opt} = \frac{1-\alpha/L}{1+\alpha/L} < 1.$$

Remarque. C'est mieux qu'avant car

$$\gamma_{\rm opt} = \frac{1-\alpha/L}{1+\alpha/L} < \gamma = \sqrt{1-\frac{\alpha^2}{L^2}}.$$

Cette vitesse de convergence est optimale pour le gradient à pas fixe: c'est le plus petit coefficient γ possible (faire l'exercice avec $J(u) = \frac{1}{2}Au \cdot u - b \cdot u$).

Vitesse de convergence optimale

Preuve. Comme $J'(u) = 0$, on écrit

$$u^{n+1} - u = u^n - u - \mu\big(J'(u^n) - J'(u)\big)$$

et on utilise un développement de Taylor avec reste exact

$$u^{n+1} - u = B^n(u^n - u) \quad \text{avec} \quad B^n = \mathrm{Id} - \mu \int_0^1 J''(u + t(u^n - u))dt.$$

Comme J' est L-Lipschitzien et J est α-convexe, on a

$$\alpha\,\mathrm{Id} \leq J'' \leq L\,\mathrm{Id} \;\Rightarrow\; (1 - \mu L)\,\mathrm{Id} \leq B^n \leq (1 - \mu\alpha)\,\mathrm{Id}.$$

Soit $\rho(B^n)$ le rayon spectral de B^n, on en déduit

$$\rho(B^n) \leq \max\left(|1 - \mu L|, |1 - \mu\alpha|\right) = \gamma < 1 \quad \text{si} \quad 0 < \mu < 2/L.$$

D'où l'on déduit $\|u^n - u\| \leq \gamma^n \|u^0 - u\|$ et l'algorithme converge.
Si on optimise le pas, on trouve $\mu_{\mathrm{opt}} = 2/(L + \alpha)$ qui conduit à

$$\rho(B^n) \leq \gamma_{\mathrm{opt}} = \max(\mu L - 1, 1 - \mu\alpha) = \frac{L - \alpha}{L + \alpha}.$$

Amélioration de la vitesse de convergence

Peut-on converger plus vite avec un algorithme d'ordre 1 ? **Oui !**

1. algorithme de Nesterov,
2. algorithme du gradient conjugué,
3. algorithme de la boule pesante.

Ces 3 algorithmes ont une meilleure vitesse de convergence, pour $\alpha/L << 1$,

$$\|u^n - u\| \leq \gamma^n \|u^0 - u\| \quad \text{avec } \gamma = 1 - \mathcal{O}(\sqrt{\frac{\alpha}{L}}).$$

Le but du jeu n'est pas d'étudier en détail ces algorithmes mais de voir leur diversité.

Remarque. La vitesse de convergence se dégrade si la fonction minimisée J est seulement convexe et pas α-convexe (voir le polycopié).

Algorithme de Nesterov

On pose $q = \alpha/L < 1$. Soit $\mu > 0$ un pas de descente.
On initialise $u^0 = u^{-1} \in V$ et $1 \leq a_0 \leq 1/\sqrt{q}$. Pour $n \geq 0$:

$$\begin{cases} a_{n+1} = \dfrac{1 - qa_n^2 + \sqrt{(1-qa_n^2)^2 + 4a_n^2}}{2} \\ v^n = u^n + \dfrac{(a_n - 1)(1 - a_{n+1}\alpha\mu)}{a_{n+1}(1-\alpha\mu)}(u^n - u^{n-1}) \\ u^{n+1} = v^n - \mu\, J'(v^n) \end{cases}$$

Idée: on extrapole u^n vers v^n avant de calculer le gradient.

Remarques.

$$\lim_{n\to+\infty} a_n = 1/\sqrt{q}$$

Si $\mu = 1/L$, le coefficient d'extrapolation qui donne v^n vérifie

$$\lim_{n\to+\infty} \frac{(a_n - 1)(1 - a_{n+1}\alpha\mu)}{a_{n+1}(1-\alpha\mu)} = \frac{1-\sqrt{q}}{1+\sqrt{q}} < 1.$$

Algorithme de Nesterov (suite)

Rappel: $q = \alpha/L < 1$, $u^0 = u^{-1} \in V$ et $1 \leq a_0 \leq 1/\sqrt{q}$.

$$\begin{cases} a_{n+1} = \dfrac{1 - qa_n^2 + \sqrt{(1-qa_n^2)^2 + 4a_n^2}}{2} \\ v^n = u^n + \dfrac{(a_n - 1)(1 - a_{n+1}\alpha\mu)}{a_{n+1}(1-\alpha\mu)}(u^n - u^{n-1}) \\ u^{n+1} = v^n - \mu\, J'(v^n) \end{cases}$$

Proposition (admise). On suppose que J est α-convexe différentiable, de dérivée L-Lipschitzienne et que $0 < \mu \leq 1/L$. Alors l'agorithme de Nesterov converge et, pour $\mu = 1/L$,

$$\|u^n - u\| \leq \frac{2}{\alpha}\gamma_{\rm Nest}^n \|u^0 - u\| \quad \text{avec} \quad \gamma_{\rm Nest} = \left(1 - \sqrt{\frac{\alpha}{L}}\right).$$

Algorithme du gradient conjugué

C'est une méthode d'ordre 1 où la direction de descente est une combinaison du gradient et de la direction de descente précédente. On introduit donc une suite supplémentaire $p^n \in V$ pour la direction de descente

$$\left\{ \begin{array}{l} u^{n+1} = u^n - \mu^n \, p^n \\ p^{n+1} = J'(u^{n+1}) + \beta^n p^n \end{array} \right.$$

où $\mu^n \in \mathbb{R}$ est le pas optimal qui minimise $\mu \to J(u^n - \mu \, p^n)$, et

$$\beta^n = \frac{\langle J'(u^{n+1}), J'(u^{n+1}) - J'(u^n) \rangle}{\|J'(u^n)\|^2}$$

C'est une méthode extrêmement populaire pour résoudre un système linéaire $Au = b$ si A est sym. déf. ≥ 0, en prenant

$$J(u) = \frac{1}{2} Au \cdot u - b \cdot u$$

Algorithme du gradient conjugué (2)

D'où vient la formule pour β^n ? Pour l'expliquer on se limite à ce cas quadratique avec $J'(u) = Au - b$.
Idée: les directions de descente doivent être conjuguées par rapport à la matrice A

$$p^{n+1} \cdot Ap^n = 0 \tag{1}$$

Lemme (voir polycopié). La formule pour β^n implique (1).

Conclusion: pour le produit scalaire $\langle u, v \rangle_A = Au \cdot v$, la suite p^n est orthogonale et les N premiers sont une base de $\mathbb{R}^N$. Or

$$J(u) = \frac{1}{2}\langle u - u^*, u - u^* \rangle_A + C \qquad \text{avec } Au^* = b$$

et on minimise successivement dans chaque direction p^n, donc l'algorithme du gradient conjugué converge exactement en N itérations.

Algorithme du gradient conjugué (3)

Proposition (admise). Soit A une matrice symétrique définie positive, de valeurs propres $0 < \lambda_1 \leq \cdots \leq \lambda_N$. Soit x la solution exacte du système $Ax = b$. Soit $(x_n)_{n\geq 0}$ la suite du gradient conjugué. Alors

$$\|x_n - x\| \leq C_0 \gamma_{GC}^n \|x_0 - x\| \qquad \text{avec } \gamma_{GC} = \frac{\sqrt{\lambda_N} - \sqrt{\lambda_1}}{\sqrt{\lambda_N} + \sqrt{\lambda_1}} \approx 1 - 2\sqrt{\frac{\lambda_1}{\lambda_N}}.$$

Remarque. Meilleure vitesse de convergence que l'algorithme de Nesterov et surtout que l'algorithme du gradient à pas fixe.

Algorithme de la boule pesante

Rappel: algorithme de gradient

$$u^{n+1} = u^n - \mu\, J'(u^n).$$

Idée: si $\mu = \Delta t$, on interprète cet algorithme comme la discrétisation en temps de l'EDO (flot gradient)

$$\dot{u}(t) = -J'(u(t))\,. \tag{1}$$

Si J a des minima locaux, une trajectoire de (1) peut y rester "coincée". Si on rajoute de **l'inertie**, on peut espérer sortir de ce puit de potentiel. On cherche donc un algorithme qui discrétise

$$m\ddot{u}(t) + \dot{u}(t) = -J'(u(t)),$$

où $m \geq 0$ serait la masse d'une boule pesante.
Pour $\mu > 0$ et $\nu > 0$, on propose

$$u^{n+1} = u^n - \mu\, J'(u^n) + \nu(u^n - u^{n-1})\,.$$

Algorithme de la boule pesante (suite)

$$u^{n+1} = u^n - \mu\, J'(u^n) + \nu(u^n - u^{n-1})\,.$$

Proposition (admise). On suppose que J est convexe, de classe C^2 tel que

$$\alpha\,\mathrm{Id} \leq J''(u) \leq L\,\mathrm{Id} \quad \text{avec} \quad 0 < \alpha \leq L\,.$$

Soit u l'unique point de minimum de J. Si u^1, u^0 sont proches de u et $0 \leq \nu < 1$ et $0 < \mu < 2(1+\nu)/L$, alors l'algorithme converge et, pour $\nu = (\frac{\sqrt{L}-\sqrt{\alpha}}{\sqrt{L}+\sqrt{\alpha}})^2$ et $\mu = 4/(\sqrt{L}+\sqrt{\alpha})^2$, on a

$$\|u^n - u\| \leq C\gamma_{BP}^n \|u^0 - u\| \quad \text{avec} \quad \gamma_{BP} = \frac{\sqrt{L}-\sqrt{\alpha}}{\sqrt{L}+\sqrt{\alpha}} \approx 1 - 2\sqrt{\frac{\alpha}{L}}.$$

Remarque. Même vitesse de convergence que l'algorithme du gradient conjugué, donc meilleure que l'algorithme du gradient à pas fixe.

Algorithme de sous-gradient

Généralisation de l'algorithme du gradient aux fonctions convexes non différentiables.

Motivation/exemple. Nombreux modèles avec fonction objectif

$$J(x) = \sup_{\lambda \in \Lambda} J_\lambda(x)$$

où, pour chaque paramètre λ, $J_\lambda(x)$ est convexe et C^1.
La fonction J n'est pas C^1 mais est convexe (exercice facile).

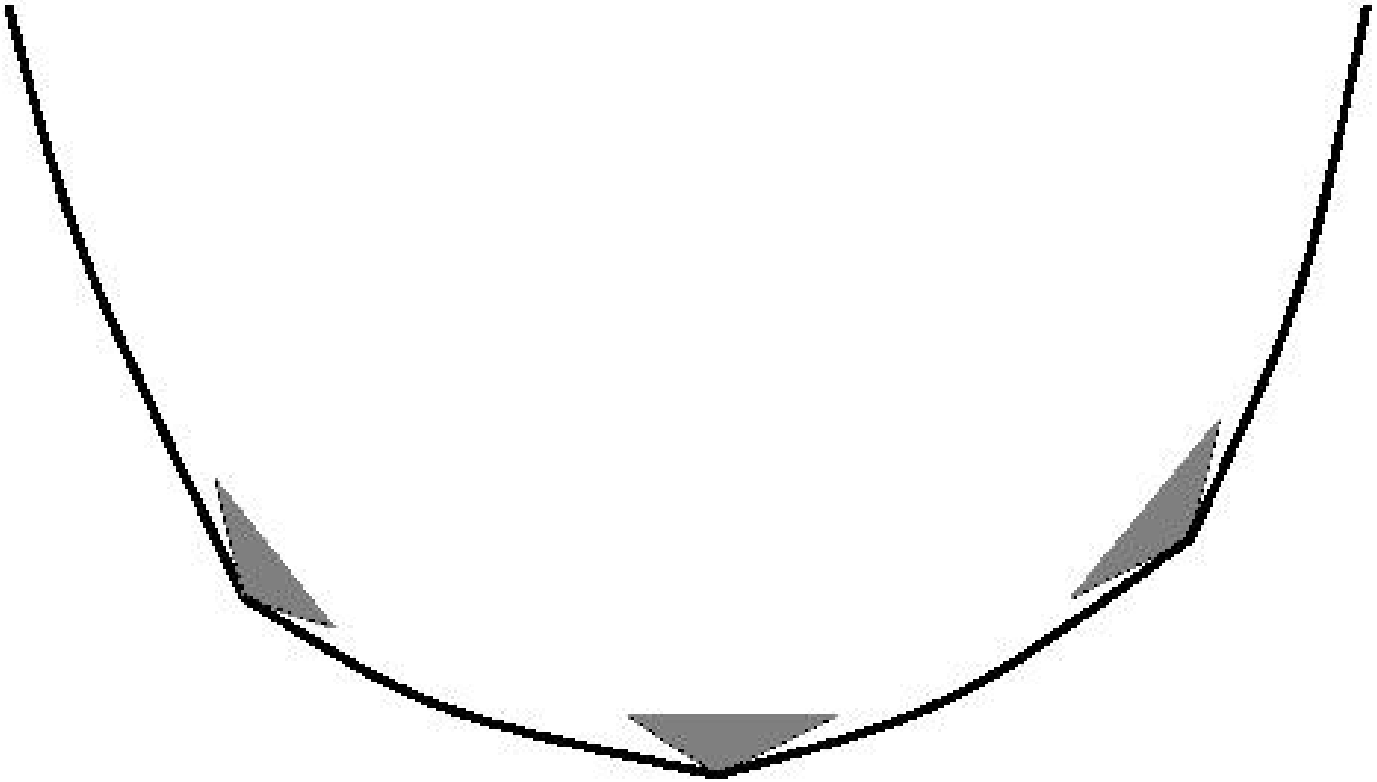
Idée fondamentale: on remplace le gradient par le **sous-gradient**.

Définition. Si J est une fonction convexe de $\mathbb{R}^N$ dans $\mathbb{R}$, on appelle sous-différentiel de J en x l'ensemble

$$\partial J(x) = \left\{ p \in \mathbb{R}^N \text{ tel que } J(y) - J(x) \geq p \cdot (y - x),\ \forall y \in \mathbb{R}^N \right\}.$$

Les éléments de $\partial J(x)$ sont appelés sous-gradients.

Exemple de fonction convexe non différentiable



Algorithme de sous-gradient (2)

Lemme. Si J est dérivable et convexe en x, alors $\partial J(x) = \{J'(x)\}$.
Preuve: écrire la convexité en x.

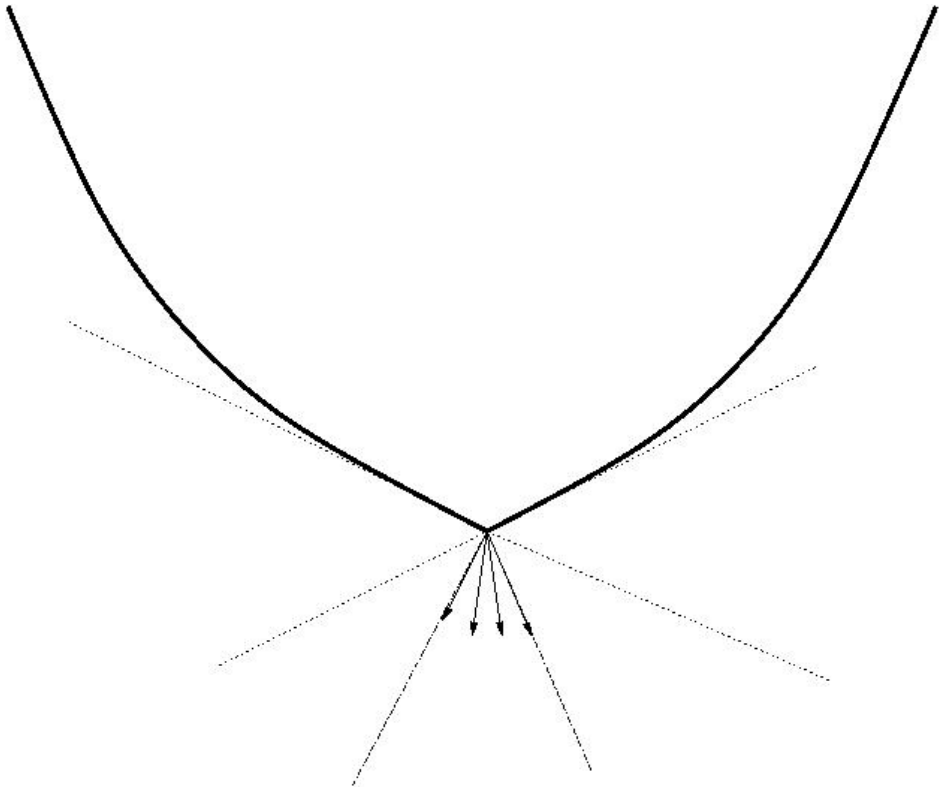
Lemme. Soit J une fonction convexe de $\mathbb{R}^N$ dans $\mathbb{R}$. Si $x \in \mathbb{R}^N$ vérifie $0 \in \partial J(x)$, alors x est un point de minimum global de J.
Preuve: d'après la définition de $0 \in \partial J(x)$, $J(y) \geq J(x)\ \forall y \in \mathbb{R}^N$.

Lemme. Soit $J(x) = \sup_{\lambda \in \Lambda} J_\lambda(x)$ avec $J_\lambda(x)$ une famille de fonctions convexes différentiables en x. Alors $J'_\lambda(x) \in \partial J(x)$ si $\lambda \in \Gamma(x) = \{\lambda \in \Lambda \text{ tel que } J_\lambda(x) = J(x)\}$.
Preuve: Comme $J_\lambda(x) = J(x)$ on a, par convexité de J_λ,

$$J(y) \geq J_\lambda(y) \geq J(x) + J'_\lambda(x) \cdot (y - x)$$

Remarque. En fait le sous-différentiel est l'enveloppe convexe des $J'_\lambda(x)$ pour $\lambda \in \Gamma(x)$.

Exemple de sous-gradient



Algorithme de sous-gradient (3)

L'**algorithme de sous-gradient** pour minimiser J convexe: pour $x_0 \in \mathbb{R}^N$, on définit

$$x_{k+1} = x_k - \frac{\rho_k}{\|p_k\|} p_k \,,$$

où $p_k \in \partial J(x^k)$ est un sous-gradient et $\rho_k > 0$ est tel que

$$\rho_k \to 0, \quad \sum_{i \in \mathbb{N}} \rho_i = +\infty, \quad \sum_{i \in \mathbb{N}} \rho_i^2 < +\infty.$$

Un exemple d'une telle suite est $\rho_k = 1/(k+1)^{1/2+\epsilon}$ avec $\epsilon > 0$. La valeur x_{k+1} n'est bien définie que si $p_k \neq 0$. Lorsque $p_k = 0$, l'algorithme s'arrête et x_k est le minimum.

Algorithme de sous-gradient (4)

$$x_{k+1} = x_k - \frac{\rho_k}{\|p_k\|} p_k$$

Remarques.

1. Le pas converge vers zéro, $\rho_k \to 0$, pour éviter des oscillations.
Cf. l'exemple $\min_{x \in \mathbb{R}} |x|$ avec une initialisation $x_0 \neq 0$.

2. Le sous-gradient est normalisé, $p_k/\|p_k\|$, car la norme d'un sous-gradient peut valoir "n'importe quoi".
Cf. l'exemple de $J(x) = |x|$ pour $x \in \mathbb{R}$ dont on vérifie que $\partial J(0) = \{p \in [-1, +1]\}$ car $|x| \geq px$ pour tout x si $-1 \leq p \leq 1$.

3. La convergence sera très lente, de l'ordre de $1/\sum_{i=1}^{k} \rho_i$, au mieux en $\mathcal{O}(k^{-1/2})$.

Algorithme de sous-gradient (5)

Proposition. Soit $J(x)$ convexe, infinie à l'infini, avec $\partial J(x) \neq \emptyset$ et localement Lipschitzienne, $\forall M > 0$, $\exists C_M > 0$ tel que

$$\|x\| + \|y\| \leq M \Rightarrow \|J(x) - J(y)\| \leq C_M \|x - y\|.$$

Alors l'algorithme du sous-gradient converge.

Preuve. Soit x_* un point de minimum de J. On développe la norme de $x_{k+1} - x_* = x_k - x_* - \rho_k p_k / \|p_k\|$

$$\begin{aligned} \|x_{k+1} - x_*\|^2 &= \|x_k - x_*\|^2 - 2\frac{\rho_k}{\|p_k\|} p_k \cdot (x_k - x_*) + \rho_k^2 \\ &\leq \|x_k - x_*\|^2 - 2\frac{\rho_k}{\|p_k\|} (J(x_k) - J(x_*)) + \rho_k^2 \end{aligned}$$

car p_k est un sous gradient de J au point x_k. On somme

$$\|x_{i+1} - x_*\|^2 + 2 \min_{0 \leq k \leq i} (J(x_k) - J(x_*)) \sum_{k=0}^{i} \frac{\rho_k}{\|p_k\|} \leq \|x_0 - x_*\|^2 + \sum_{k=0}^{i} \rho_k^2.$$

Donc la suite x_i est bornée.

Algorithme de sous-gradient (6)

$$\|x_{i+1}-x_*\|^2+2\min_{0\le k\le i}(J(x_k)-J(x_*))\sum_{k=0}^{i}\frac{\rho_k}{\|p_k\|}\le\|x_0-x_*\|^2+\sum_{k=0}^{i}\rho_k^2.$$

Comme la suite x_k est bornée et J est Lipschitzienne, on obtient $\|p_k\|\le C_M$ (voir le polycopié) et on en déduit

$$\min_{0\le k\le i}(J(x_k)-J(x_*))\le\frac{C_M}{2}\frac{\|x_0-x_*\|^2+\sum_{k=0}^{\infty}\rho_k^2}{\sum_{k=0}^{i}\rho_k}$$

Comme la série des ρ_k diverge, on en déduit la convergence de $J(x_k)$ vers $J(x_*)$ (et de x_k vers x_*, voir le polycopié).

Remarque.
La convergence est très lente puisque pour $\rho_k=1/(k+1)^{1/2+\epsilon}$ avec $\epsilon>0$ on trouve une vitesse en $k^{-1/2+\epsilon}$, ce qui est plus lent que la convergence géométrique pour l'algorithme du gradient.

Gradient stochastique

Motivation: apprentissage machine en dimension n très grande.

$$\inf_{x\in\mathbb{R}^d} F(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x).$$

Algorithme du gradient stochastique. Initialisation $x^0 \in \mathbb{R}^d$.
Pour $k \geq 1$:

$$x^{k+1} = x^k - \mu^k f_{i_k}'(x^k),$$

avec un pas de descente $\mu^k > 0$ et un indice i_k tiré aléatoirement et uniformément dans l'ensemble $\{1, ..., n\}$.

Intérêt: une itération de l'algorithme est très économique (et assez insensible aux erreurs numériques).

Variante: algorithme de mini-batch de taille m.

$$x^{k+1} = x^k - \mu^k \frac{1}{m}\sum_{i\in\mathcal{I}_k} f_i'(x^k),$$

$\mathcal{I}_k$ une collection de m indices distincts aléatoires dans $\{1, ..., n\}$.

Gradient stochastique (2)

Proposition. On suppose que $F(x)$ est α-convexe, de minimum x^* et qu'il existe $C > 0$, indépendante de n, telle que

$$\frac{1}{n}\sum_{i=1}^{n} \|f_i'(x)\|^2 \leq C(1 + \|x - x^*\|^2) \quad \forall x \in \mathbb{R}^d.$$

Si la suite des pas est $\mu^k = \frac{1}{k+1}$, alors l'algorithme du gradient stochastique converge.

Preuve. On calcule la norme au carré de

$$x^{k+1} - x^* = x^k - x^* - \mu^k f_{i_k}'(x^k),$$

$$\|x^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - 2\mu^k (x^k - x^*) \cdot f_{i_k}'(x^k) + (\mu^k)^2 \|f_{i_k}'(x^k)\|^2,$$

dont on prend l'espérance en la variable aléatoire de l'indice i_k

$$\mathbb{E}\left(\|x^{k+1} - x^*\|^2\right) = \|x^k - x^*\|^2 - 2\mu^k (x^k - x^*) \cdot F'(x^k) + \frac{(\mu^k)^2}{n}\sum_{i=1}^{n} \|f_i'(x^k)\|^2$$

car $\mathbb{E}(f_{i_k}'(x^k)) = F'(x^k)$.

Gradient stochastique (3)

Forte convexité $(x^k - x^*) \cdot (F'(x^k) - F'(x^*) \geq \alpha \|x^k - x^*\|^2$, donc

$$\|x^{k+1} - x^*\|^2 \leq \left(1 - 2\mu^k \alpha\right) \|x^k - x^*\|^2 + \frac{(\mu^k)^2}{n} \sum_{i=1}^{n} \|f_i'(x^k)\|^2.$$

Utilisant l'hypothèse sur les dérivées, il vient

$$\|x^{k+1} - x^*\|^2 \leq \left(1 - 2\mu^k \alpha + C(\mu^k)^2\right) \|x^k - x^*\|^2 + C(\mu^k)^2.$$

On choisit $0 < \mu^k < 2\alpha/C$ pour avoir

$$0 < \rho^k = 1 - 2\mu^k \alpha + C(\mu^k)^2 < 1.$$

En posant $\Pi^k = \prod_{j=0}^{k} \rho^j$ on en déduit

$$\|x^{k+1} - x^*\|^2 \leq \Pi^k \|x^0 - x^*\|^2 + C\Pi^k \sum_{i=0}^{k} \frac{(\mu^i)^2}{\Pi^i}$$

On conclut à la convergence avec $\mu^k = 1/(k+1)$ car

$$\rho^k = 1 - \frac{2\alpha}{k+1} + \mathcal{O}(k^{-2}),\ \Pi^k = \mathcal{O}(k^{-2\alpha}),\ \Pi^k \sum_{i=0}^{k} \frac{(\mu^i)^2}{\Pi^i} = \mathcal{O}(k^{-1}).$$

Algorithme proximal

Motivation: moindres carrés et parcimonie.
Pour expliquer des résultats $b \in \mathbb{R}^p$ à partir de données A matrice $p \times n$, on cherche un modèle linéaire de paramètres $x \in \mathbb{R}^n$

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|^2.$$

Classiquement, il existe une solution optimale x^*, solution de

$$A^*Ax^* = A^*b \quad \text{unique si } \ker A = \{0\}.$$

Parcimonie: on peut vouloir aussi minimiser le nombre de paramètres explicatifs (avec $\tau > 0$)

$$\min_{x \in \mathbb{R}^n} \frac{1}{2\tau}\|Ax - b\|^2 + \|x\|_1 \quad \text{avec} \quad \|x\|_1 = \sum_{j=1}^{n} |x_j|.$$

Problème de statistique appelé LASSO (least absolute shrinkage and selection operator).

Pourquoi la minimisation de la norme ℓ^1 diminue le nombre de composantes non nulles de x ?

Explication dans un cas simple: soit $\tau > 0$ et $p = n$

$$\min_{x \in \mathbb{R}^n} \frac{1}{2\tau}\|x - b\|^2 + \|x\|_1 \quad \text{avec} \quad \|x\|_1 = \sum_{j=1}^{n} |x_j|.$$

Soit x^* le minimum unique

$$\begin{array}{l} \text{soit } x_i^* = 0 \\ \text{soit } x_i^* > 0 \text{ et } x_i^* - b_i + \tau = 0 \\ \text{soit } x_i^* < 0 \text{ et } x_i^* - b_i - \tau = 0 \end{array}$$

On obtient l'opérateur de contraction (shrinkage, en anglais)

$$x_i^* = S_\tau(b_i) = \begin{cases} 0 & \text{si } |b_i| < \tau \\ b_i - \operatorname{sgn}(b_i)\tau & \text{si } |b_i| \geq \tau \end{cases}$$

Algorithme proximal (2)

Plus généralement, on considère

$$\min_{x\in\mathbb{R}^n} \frac{1}{\tau} J_1(x) + J_2(x)$$

avec J_1 différentiable et J_2 convexe mais non différentiable.

Exemple: $J_1(x) = \frac{1}{2}\|Ax - b\|^2$ et $J_2(x) = \|x\|_1$.

Pour J_1, plein d'algorithmes de type gradient !
Pour J_2, algorithme de sous-gradient ou bien... algorithme proximal à définir.

Séparation d'opérateurs: un algorithme pour J_1, un autre pour J_2.

Algorithme proximal (3)

Idée: faire un algorithme de gradient implicite

$$x^{k+1} = x^k - \tau J_2'(x^{k+1}) \tag{1}$$

C'est "conceptuel" car il faut résoudre une équation en x^{k+1}...
En plus, il faudrait que J_2 soit différentiable...
Mais (1) correspond à x^{k+1} minimum de

$$\min_{x\in\mathbb{R}^n} \frac{1}{2\tau}\|x - x^k\|^2 + J_2(x) \tag{2}$$

On dit que (1) ou (2) définit l'application proximale de τJ_2

$$\mathrm{prox}_{\tau J_2}(x^k) = x^{k+1}$$

Pour $J_2(x) = \|x\|_1$ (convexe, non différentiable) on sait calculer exactement $x^{k+1} = \mathrm{prox}_{\tau J_2}(x^k) = S_\tau(x^k)$!

Algorithme proximal (4)

Algorithme de type explicite-implicite: initialisation $x^0 \in \mathbb{R}^n$,

$$\left\{ \begin{array}{l} x^{k+1/2} = x^k - \tau J_1'(x^k) \\ x^{k+1} = \mathrm{prox}_{\tau J_2}(x^{k+1/2}) \end{array} \right.$$

Pour le cas particulier

$$\min_{x \in \mathbb{R}^n} \frac{1}{2\tau} \|Ax - b\|^2 + \|x\|_1 \quad \text{avec} \quad \|x\|_1 = \sum_{j=1}^{n} |x_j|.$$

cet algorithme proximal est

$$x^{k+1} = S_\tau \Big(x^k - \tau A^*(Ax^k - b) \Big)$$

Remarque. Cet algorithme converge... mais c'est une autre histoire.