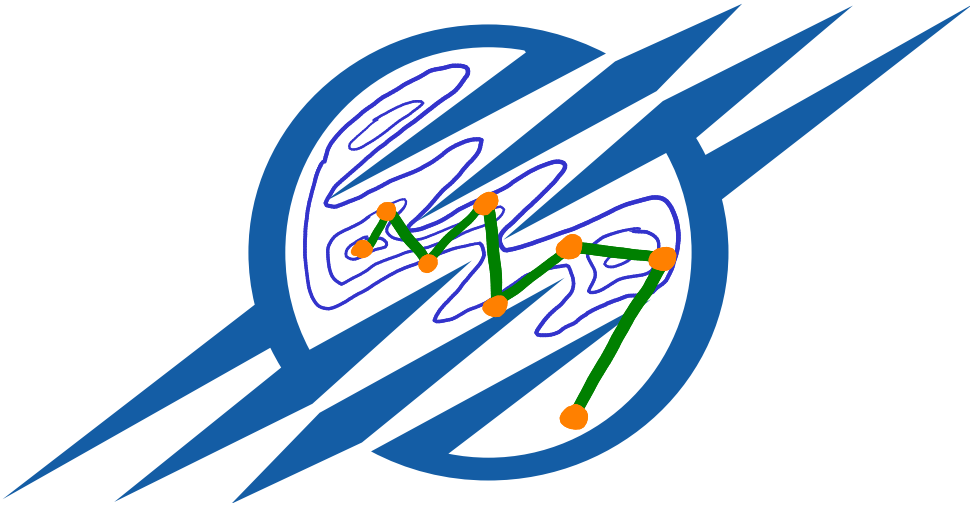
Quelques notes illustrées d'optimisation convexe

document préliminaire – à compléter

Antoine Tilloy
CAS – Mines Paris PSL
antoine.tilloy@minesparis.psl.eu



Ces notes sont la version bande dessinée du polycopié d'optimisation [1] *pour le cours ECUE 21.1 écrit par Delphine Bresch-Pietri et Nicolas Petit. En amateur, on se concentre sur éléments intuitif et visuels, sans prouver les théorèmes (dont on reprend la numérotation). À l'examen, seul le polycopié fait foi.*

1. Delphine Bresch-Pietri and Nicolas Petit. Optimisation, 2022

Pourquoi l'optimisation convexe ?

Tout problème mathématique raisonnable peut être reformulé comme un problème de minimisation d'un coût. Par exemple, si on cherche à trouver $x \in \mathbb{R}^n$ tel que $A(x) = B(x)$ où $A, B : \mathbb{R}^n \to \mathbb{R}^m$ sont des fonctions compliquées, on peut trouver x^* solution (à supposer qu'elle existe) en minimisant la norme de la différence

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^n} \|A(x) - B(x)\|^2 . \tag{1}$$

Évidemment rien n'est gratuit : il existe des problèmes mathématiques dont on peut prouver qu'ils sont difficiles, et par conséquent l'optimisation d'une fonction générique est en général très difficile. Pour reprendre Yurii Nesterov [2] « les problèmes d'optimisation sont en général insolubles ». En effet, pour les problèmes intéressants n est grand et dans les cas méchants, on peut avoir un nombre exponentiel en n de minima locaux qui va rendre la minimisation globale infaisable.

2. Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003

Il faut donc trouver un cadre un peu restrictif, dans lequel on peut développer une théorie de l'optimisation, où optimiser une fonction est faisable en général. Aujourd'hui, on pense que le bon cadre est celui des fonctions convexes minimisées sur un ensemble convexe, a.k.a. l'optimisation convexe. C'est un cadre ni trop trivial ni trop difficile, où on peut prouver des choses et où on peut construire des algorithmes efficaces. De plus, même si dans la vraie vie les fonctions sont rarement convexes, les techniques que l'on développe en optimisation convexe tendent à fonctionner aussi dans le monde réel (entre autre parce que les fonctions d'intérêt sont en général au moins convexes près du minimum).

1
Généralités théoriques

On se demande ici dans quels cas une fonction a un minimum (local ou global), et on regarde plus particulièrement le cas des fonctions convexes (assez général pour être intéressant, pas trop pour rester soluble).

Un bon complément pour qui veut explorer ces aspects théoriques est le cours de Stanford :
Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. URL https://web.stanford.edu/~boyd/cvxbook/

1.1 *Conditions d'optimalité*

1.1.1 *Pour une fonction générique*

Que peut-on dire sur l'existence d'un minimum quand on ne sait presque rien sur f (c'est à dire même pas qu'elle est convexe) ni de l'ensemble sur lequel on la minimise ?

Théorème 1 (Weierstrass). *f continue sur K compact $\implies$ f admet un minimum*

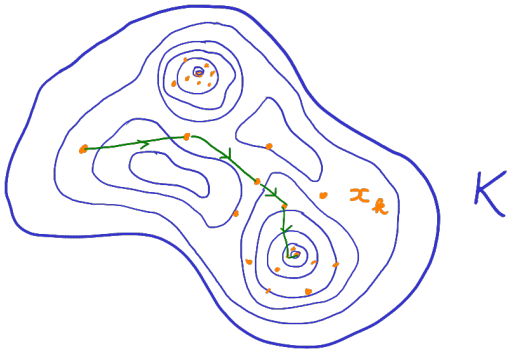


FIGURE 1.1: En bleu, les lignes de niveau de f, en orange, une suite x_n telle que $f(x_k)$ s'approche de $\inf f$ qui est manifestement approché en deux points. On extrait une sous-suite $\tilde{x}_n$ en vert qui converge et donc force à aller vers un minimum.

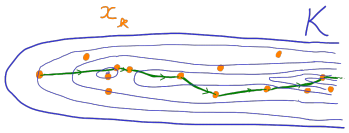


FIGURE 1.2: Si K n'est pas compact, on ne peut pas extraire une sous-suite convergente, car l'infimum peut être atteint à l'infini. Par exemple la fonction exponentielle n'a pas de minimum sur $\mathbb{R}$.

Idée de preuve. On utilise le fait que f étant continue sur un compact elle est bornée puis

1. on prend une suite x_n telle que $f(x_k) \underset{k\to+\infty}{\longrightarrow} \inf f$
2. on extrait une sous suite $\tilde{x}_n$ convergente (voir Fig. 1.1)

Théorème 2. *$f : \mathbb{R}^n \to \mathbb{R}$ continue et $f(x) \underset{\|x\|\to+\infty}{\longrightarrow} +\infty$ $\implies$ f admet un minimum*

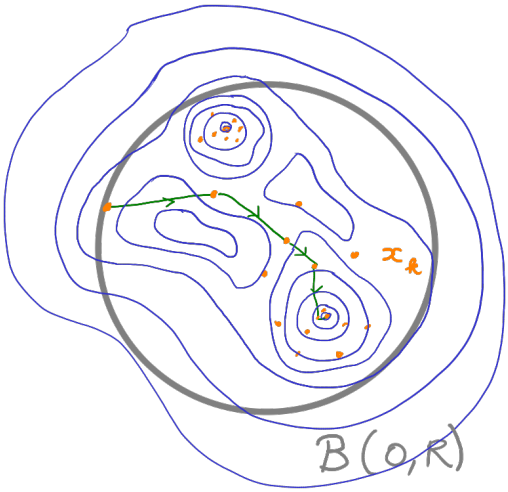


FIGURE 1.3: Si K n'est pas compact, alors en se restreignant à une boule de rayon suffisamment grand, on se ramène au cas compact

Idée de preuve. f est énorme hors d'une boule $B(0,R)$ pour un rayon R assez grand (voir Fig. 1.3), donc on peut se ramener à la minimisation à l'intérieur de cette boule et donc au théorème précédent

1.1.2 *Pour une fonction deux fois différentiable*

On considère maintenant une fonction deux fois différentiable, ayant donc un gradient ∇f et une Hessienne [1] $\nabla^2 f$ bien définis.

Théorème 3 (Condition nécessaire de minimalité). *x^* minimiseur local de $f \in \mathscr{C}^2(\mathbb{R}^n) \implies \nabla f(x^*) = 0$ et $\nabla^2 f(x^*) \succeq 0$*

Théorème 4 (Condition suffisante de minimalité). *$\nabla f(x^*) = 0$ et $\nabla^2 f(x^*) \succ 0 \implies x^*$ minimiseur local de f*

Idée de preuve. Dans les deux cas, on le voit en faisant un développement limité de f autour de x^*

$$f(x^*+h) = f(x^*) + h^T \nabla f(x^*) + \frac{1}{2} h^T \nabla^2 f(x^*) h + o(\|h\|^2). \quad (1.1)$$

Évidemment on ne peut rien dire sur le caractère global du minimum sans plus d'information sur f !

1. On note ici $\nabla^2 f$ le Hessien (ou la matrice Hessienne) de f, *i.e.* la matrice $[\nabla^2 f(x)]_{ij} := \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$. Il ne s'agit pas du Laplacien Δ (noté aussi parfois ∇^2) qui en est seulement la trace.

FIGURE 1.4: La stricte positivité de la Hessienne n'est pas une condition nécessaire de minimalité, par exemple :

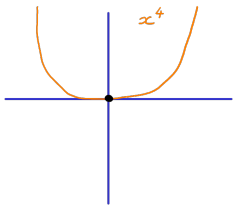
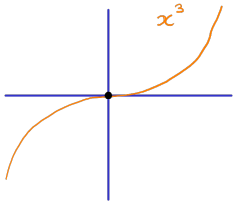


FIGURE 1.5: La positivité (non-négativité) de la Hessienne n'est pas une condition suffisante de minimalité, par exemple :



1.2 *Fonctions convexes*

1.2.1 *Généralités*

Une fonction convexe est simplement une fonction qui est au dessous de toutes ses cordes (sur E convexe) [2]

$$\forall x, y \in E, \forall \lambda \in [0,1],\ f(\lambda x + (1-\lambda) y) \leq \lambda f(x) + (1-\lambda) f(y) \quad (1.2)$$

Cette définition peut être reformulée de deux manières différentes avec le théorème suivant.

2. La définition de convexité n'a de sens que si l'ensemble E est convexe. Même si $f : x \mapsto x^2$ est manifestement convexe sur $\mathbb{R}$ ou $[0,1]$, on ne peut même pas vérifier la définition si $E = [-1,0] \cup [1,+\infty[$ qui n'est pas convexe.

Théorème 5. *E convexe d'intérieur non vide de $\mathbb{R}^n$ et $f : E \to \mathbb{R}$ continue (et on suppose*[3] *$\mathscr{C}^1$)*

1. *f est convexe [avec la définition (1.2)]*
2. *L'épigraphe de f, $Epi(f)$, est convexe*
3. *L'hyperplan d'appui de f en x est sous f pour tout x*

3. Si f n'est pas dérivable, il faut remplacer l'« hyperplan d'appui » par « les hyperplans d'appui ».

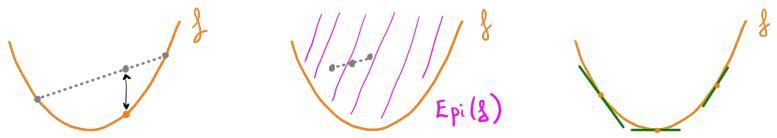


FIGURE 1.6: Trois caractérisations de la convexité : par la définition (1.2), par la convexité de l'épigraphe, et par les hyperplans d'appui en chaque point.

Une fonction convexe n'est pas forcément différentiable, mais on peut définir un sous-gradient $v \in \mathbb{R}^n$ en x_0 qui est la direction possible d'une droite passant par $x_0, f(x_0)$ et restant sous f

$$\forall x \in E,\ f(x) \geq f(x_0) + v^T(x - x_0) \tag{1.3}$$

Le sous-différentiel $\partial f(x_0)$ en x_0 est l'ensemble des sous-gradients en x_0 (voir Fig. 1.7). Si f est régulière, alors le sous-gradient se réduit à un point, qui est le gradient $\partial f(x_0) = \{\nabla f(x_0)\}$.

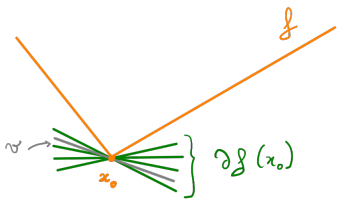


FIGURE 1.7: Le sous-differentiel $\partial f(x_0)$ en un point où f n'est pas dérivable, avec un exemple de sous-gradient v.

1.2.2 *Lien convexité et minima*

On a la tautologie suivante :

Théorème 6. *Soit $f : \Omega \to \mathbb{R}$ convexe.*
Alors x^* est minimiseur global *de $f \Leftrightarrow\ 0 \in \partial f(x_0)$*

Théorème 7. *Soit f convexe, tout minimum local de f est global*

Si f est régulière ($\mathscr{C}^1$ ou $\mathscr{C}^2$ pour le 3e point), on a encore d'autres caractérisations des fonctions convexes

Théorème 9. *f est convexe sur $\Omega \subset \mathbb{R}^n$ est équivalent à*

1. *f au dessus des tangentes $\forall x, y \in \Omega, f(y) \geq f(x) + \nabla f(x)^T(y - x)$*
2. *∇f monotone : $\forall x, y \in \Omega, [\nabla f(x) - \nabla f(y)]^T(x - y) \geq 0$*
3. *La matrice Hessienne est positive partout $\forall x \in \Omega,\ \nabla^2 f(x) \succeq 0$*

Théorème 10. *Soit $f \in \mathscr{C}^1(\Omega \subset \mathbb{R}^n)$ convexe*
x^* minimiseur global sur $\Omega \Leftrightarrow\ \nabla f(x^*) = 0$

Idée de preuve. C'est une conséquence immédiate de l'équivalence entre convexité et être au dessus de ses tangentes.

FIGURE 1.8: Situation du théorème 10

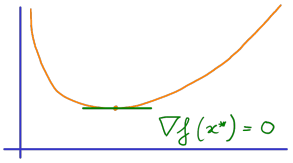
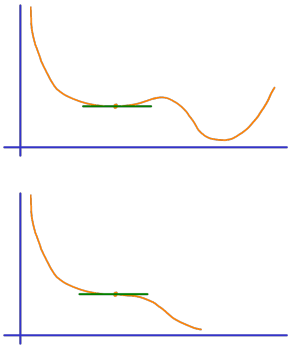


FIGURE 1.9: Nécessité de la convexité dans le théorème 10 pour avoir un minimum global. Sans convexité on peut avoir un minimum local ou même pas de minimum du tout.



1.2.3 *Renforcement de la notion de convexité*

La fonction $x \mapsto |x|$ est convexe avec les définitions précédentes, mais on sent qu'elle l'est tout juste ; elle n'est pas convexe au sens "commun" d'une courbe bombée. On peut renforcer la notion de

convexité de deux manières, avec la convexité *stricte* et la convexité *forte* (α-convexité).

La convexité stricte revient simplement à mettre une inégalité stricte dans la définition (en interdisant à λ de toucher le bord) :

$$\forall x, y \in E, \forall \lambda \in]0,1[,\ f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda) f(y).$$

La convexité forte revient à demander une marge α de positivité aux valeurs propres de la matrice Hessienne en chaque point, *i.e.* $\lambda_i(x) \geq \alpha$ où $\lambda_i(x)$ est la i-ème valeur propre de $\nabla^2 f(x)$. On peut le traduire sans faire référence explicitement à la Hessienne : f est fortement convexe s'il existe $\alpha > 0$ tel que la fonction $x \mapsto f(x) - \frac{\alpha}{2}\|x\|^2$ est convexe.

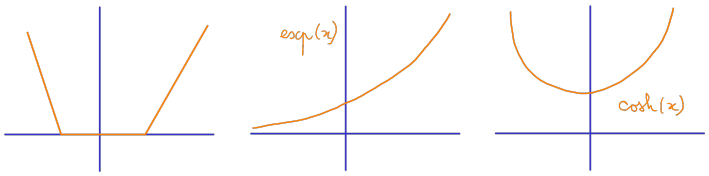


FIGURE 1.10: Une fonction convexe, strictement convexe, et fortement convexe

Théorème 12. *Soit $f : \mathbb{R}^n \to \mathbb{R}$ fortement convexe et différentiable, alors f admet un unique minimum global sur tout fermé convexe de $\mathbb{R}^n$*

Intuitivement, demander la forte convexité évite les deux premières situations de la figure 1.10.

2
Méthodes pour l'optimisation sans contrainte

On se demande comment minimiser en pratique une fonction suffisamment régulière $f : \mathbb{R}^n \to \mathbb{R}$, idéalement fortement convexe, avec des méthodes de descente.

2.1 *Quelques généralités*

2.1.1 *Le cas réel est très simple*

Minimiser une fonction convexe d'une seule variable est trivial, et peut se faire par dichotomie, qui converge exponentiellement vite vers la solution $\to$ numériquement trivial. Mais c'est une stratégie qui ne se généralise pas à $f : \mathbb{R}^n \to \mathbb{R}$. Pour la suite, il faut imaginer n grand, et en tout cas $n \geq 2$ pour que la situation soit suffisamment non-triviale pour se forger une intuition.

2.1.2 *Méthode de descente*

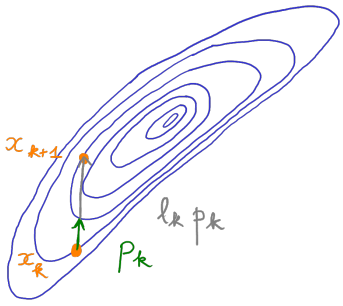
Pour trouver le minimum d'une fonction convexe, l'idée générale est de commencer par un x_0 aléatoire et de procéder de manière itérative en déplaçant x_k à chaque étape k de manière à réduire $f(x_k)$. Cela demande deux choix :

1. une direction de descente p_k,
2. un pas ℓ_k, qui détermine à quel point on va loin.

$$x_{k+1} = x_k + \overbrace{\ell_k}^{\text{pas}} \underbrace{p_k}_{\text{direction}} \tag{2.1}$$

Commençons par discuter le problème de choix de pas sur la direction la plus simple.

FIGURE 2.1: Direction p_k et pas l_k



Remarque. Le cas réel est trivial, la direction n'est qu'un signe, et ainsi tous les choix de descente qui réduisent f sont équivalents à un changement de pas.

2.2 *Descente de gradient et choix du pas*

2.2.1 *Idée*

Le plus simple est de prendre $p_k = -\nabla f(x_k)$, qui consiste à déplacer x_k dans la direction de plus grande pente. Difficile d'imaginer plus naturel !

Le pas le plus simple est ensuite le pas constant $\ell_k \equiv \ell$. Typiquement, il peut être (voir Fig. 2.2)

— trop petit, on a besoin d'un nombre d'itérations colossal pour arriver au minimum

— trop grand, la minimisation est instable et la fonction diverge

— de la bonne taille, quand on a de la chance...

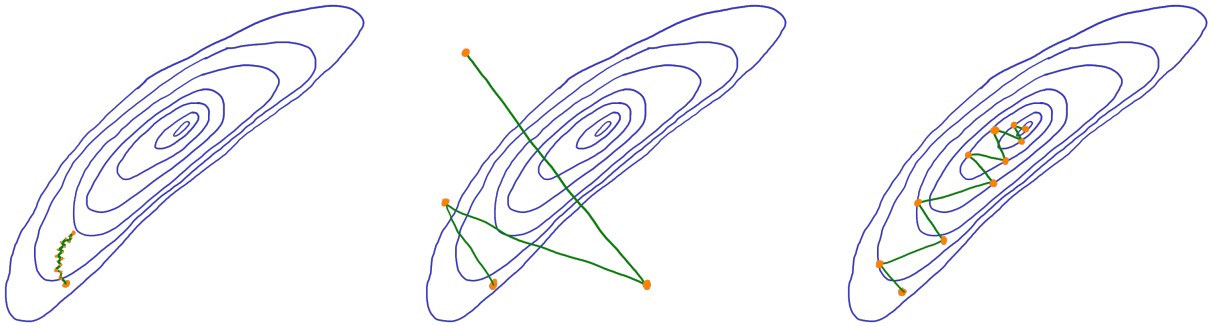


FIGURE 2.2: Descente de gradient avec un pas trop petit, trop grand, et optimal.

Pour éviter de choisir le pas au hasard, et de tomber presque sûrement dans le premier ou le deuxième cas, l'idéal serait de choisir à chaque étape le *pas optimal* :

$$\ell_k^* := \underset{\ell \in \mathbb{R}^+}{\operatorname{argmin}} f(x_k + \ell p_k)\,. \tag{2.2}$$

C'est de nouveau un problème d'optimisation, mais cette fois-ci beaucoup plus simple car sur $\mathbb{R}^+$ (voir Fig. 2.3). On minimise en effet

$$g : \ell \mapsto f(x_k + \ell p_k) \tag{2.3}$$

dont la dérivée (la pente) vaut $g'(\ell) = \nabla f(x_k + \ell p_k)^T p_k$. En général, on n'a pas envie de passer trop de temps à trouver le pas optimal, et trouver un pas correct suffit. Mais comment savoir qu'un pas est correct ?

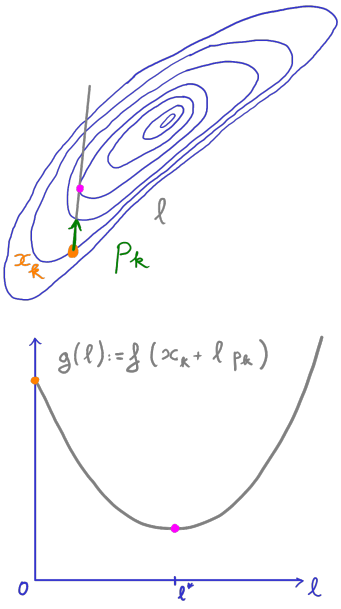


FIGURE 2.3: La minimisation le long de la direction de descente est un problème unidimensionnel.

2.2.2 *Conditions de Wolfe*

La première est la *condition d'Armijo*, qui est une sorte de condition anti-rebond, et permet ainsi de s'assurer qu'on ne pas fait pas un pas trop grand. Elle s'écrit pour $1 > c_1 > 0$ et une direction de descente p_k [qui n'est pas forcément $-\nabla f(x_k)$] :

$$f(x_k + \ell_k p_k) \leq f(x_k) + c_1\, \ell_k \underbrace{\nabla f(x_k)^T p_k}_{\leq 0}. \tag{2.4}$$

Intuitivement, si $c_1 \to 0$ on est très tolérant, on accepte tous les pas tant que f ne remonte pas plus haut que son point de départ. Si $c_1 \to 1$, on est hyper strict, on n'accepte que des pas très petits pour être sûr de ne pas rebondir.

La seconde est la *condition de courbure*. Intuitivement, elle empêche de prendre des pas trop petits en demandant que la pente

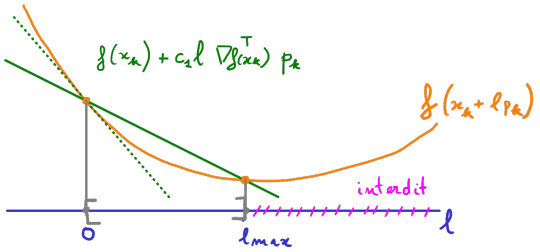


FIGURE 2.4: Condition d'Armijo : interdit les ℓ trop grands

de f dans la direction p_k ait substantiellement changé pendant la descente (sans quoi on pouvait continuer à descendre et prendre un pas plus grand). Formellement, pour $1 > c_2 > 0$, elle s'écrit

$$\underbrace{-\nabla f(x_k + \ell p_k)^T p_k}_{\geq 0} \leq \underbrace{-c_2 \nabla f(x_k)^T p_k}_{\geq 0} \tag{2.5}$$

Intuitivement, si $c_2 \to 1$, on est très tolérant, et on considère que tous les pas $\ell \leq 0$ sont suffisamment grands. À l'inverse, si $c_2 \to 0$, on est exigent et veut un pas suffisamment grand pour que la pente s'annule, *i.e.* que f remonte.

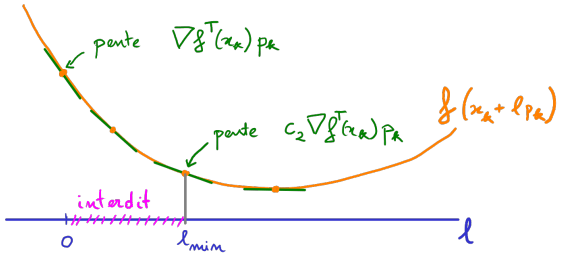


FIGURE 2.5: Condition de courbure : interdit les ℓ trop petits

On peut combiner la condition de courbure et la condition d'Armijo et avoir un pas ni trop grand ni trop petit. Cette combinaison s'appelle les conditions de Wolfe (voir Fig. 2.6), et elles peuvent être satisfaites si et seulement si $1 > c_2 > c_1 > 0$.

Théorème 13. *Les conditions de Wolfe admettent un intervalle de solution* $[\ell_{min}, \ell_{max}]$ *lorsque* $1 > c_2 > c_1 > 0$

Intuitivement au moins, c'est logique, si $c_2 \leq c_1$, les deux conditions sont exigeantes et demandent ainsi des tailles de pas incompatibles (Armijo veut très petit, courbure veut très grand). Si $c_1 \leq c_2$, on imagine que l'on doit pouvoir satisfaire les deux en même temps, ce que l'on peut prouver. L'intérêt d'un pas vérifiant les conditions de Wolfe est double. Il permet de garantir *en théorie* que les algorithmes de descente se comportent bien, et il permet *en pratique* d'accélérer la convergence. Pour trouver un pas vérifiant les conditions de Wolfe, on peut par exemple partir d'un pas très grand, vérifiant les conditions de courbure, qu'on diminue ensuite jusqu'à ce qu'il vérifie la condition d'Armijo.

En pratique, un bon choix de c_1, c_2 peut dépendre (même si faiblement) de la direction de descente (quasi Newton ou gradient conjugué). Généralement, on prend des conditions de Wolfe « tolérantes » pour ne pas passer trop de temps à raffiner le pas. Par exemple, `Scipy` utilise par défaut $c_1 = 10^{-4}$ et $c_2 = 0.9$ (voir `scipy.optimize.line_search`) comme le suggèrent aussi Nocedal et Wright.

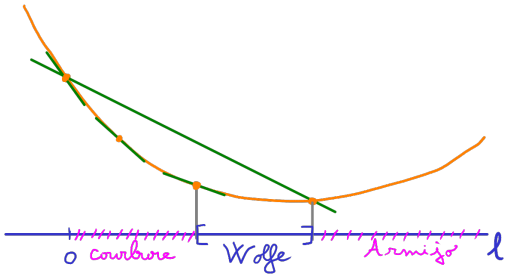


FIGURE 2.6: Conditions de Wolfe : interdit les ℓ trop petits avec la condition de courbure et les pas trop grand avec la condition d'Armijo, laissant un intervalle de pas autorisés pour $1 > c_2 > c_1 > 0$.

2.2.3 *Limites de la descente de gradient*

L'idée de descendre le long de la plus grande pente, c'est à dire orthogonalement aux lignes de niveaux semble évidente, et pourtant elle n'est pas optimale, ce qui est particulièrement visible lorsque les lignes de niveaux sont très aplaties (voir Fig. 2.7).

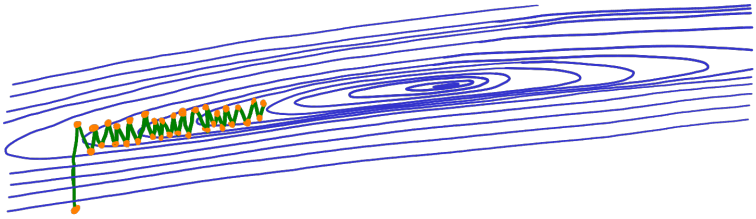


FIGURE 2.7: Descente de gradient pour une fonction f avec $\nabla^2 f(x^*)$ mal conditionné (donnant des lignes de niveau en ellipse très aplatie). Sur ce dessin, l'algorithme convergerait toujours en un nombre d'itérations raisonnable, mais il faut imaginer qu'en pratique l'ellipse peut être un million de fois plus aplatie, et alors la convergence apparaît bloquée.

Le phénomène d'oscillation de la figure 2.7 se traduit par un ralentissement de la convergence, visible dans la meilleure borne que l'on peut prouver analytiquement pour une fonction quadratique

$$f : x \mapsto \frac{1}{2} x^T Q x - bx \tag{2.6}$$

où $Q \succ 0$. Dans ce cas, on peut montrer que la descente de gradient à *pas optimal* converge et que l'on a

$$\|x_{k+1} - x^*\|_Q \leq \frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1} \|x_k - x^*\|_Q \tag{2.7}$$

où $\|x\|_Q^2 := x^T Q x$ (la norme associée à Q) et λ_n, λ_1 sont la plus grande et la plus petite valeur propre de Q. On voit ainsi que quand Q est mal conditionnée, *i.e.* que les lignes de niveau forment un ellipsoïde très aplati, *i.e.* $\lambda_n / \lambda_1 \gg 1$, la convergence est très lente [1] (même pour un pas optimal).

Numériquement, l'optimisation par descente de gradient d'une fonction avec une Hessienne très mal conditionnée (par exemple avec un ratio de valeurs propres $\frac{\lambda_2}{\lambda_1} = 10^9$) peut donner l'impression de se bloquer à des valeurs dépendant de l'initialisation, comme s'il y avait des minima locaux. Descendre dans la vallée est quasi instantané, on y arrive en un point dépendant des conditions initiales. Ensuite, descendre le long de la vallée se fait à coût d'oscillations infinitésimales qui laissent la fonction quasi constante.

1. Techniquement, cela montre uniquement que la meilleure inégalité que l'on peut prouver implique une convergence lente. Mais, en pratique, dans une situation générique, cette borne donne une bonne idée de la convergence

2.3 *Méthode de Newton*

Pour trouver une meilleure direction, on commence par considérer de nouveau une fonction quadratique $f : x \mapsto \frac{1}{2} x^T Q x - bx$ avec Q symétrique définie positive. Pour une telle fonction, on peut calculer le gradient et l'annuler explicitement :

$$\nabla f(x) = 0 \implies Qx = b \implies x = Q^{-1} b\,. \tag{2.8}$$

Ainsi, si f est quadratique, on peut la minimiser en une itération en partant de $x = 0$ et en descendant dans la direction $p = Q^{-1}b$ avec un pas $\ell = 1$. En général $f \in \mathscr{C}^2(\mathbb{R}^n)$ n'est pas quadratique mais elle est au moins localement quadratique au voisinage du point x_k obtenu à la précédente itération et on a

$$f(x_k + \ell p_k) = f(x_k) + \ell \nabla f(x_k)^T p_k + \frac{\ell^2}{2} p_k^T \nabla f^2(x_k) p_k + o(\ell^2)$$

On peut désormais minimiser f en fonction de ℓp_k en l'approximant comme une fonction quadratique, ce qui correspond à prendre $Q = \nabla^2 f(x_k)$ et $b = -\nabla f(x_k)$ et donne

$$p_k = -\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k) \tag{2.9}$$

avec un pas $\ell \equiv 1$. C'est l'algorithme de Newton. Il donne essentiellement la meilleure direction de descente possible en tenant compte des dérivée secondes. Évidemment, la méthode est beaucoup plus coûteuse par itération, en demandant le calcul de la matrice Hessienne $\nabla^2 f(x_k)$ et son inversion, ce qui a génériquement un coût $\propto n^3$ *par itération* et est prohibitif pour des problèmes de grande taille.

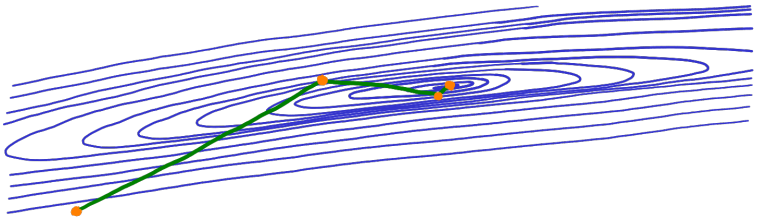


FIGURE 2.8: Convergence de la méthode de Newton vers le minimum. Si les lignes de niveau formaient exactement une ellipse (fonction quadratique), la méthode convergerait en une seule itération. Dans le cas général, il faut plusieurs itérations pour s'approcher du minimum mais les oscillations de la descente de gradient sont supprimées, même pour des lignes de niveau très aplaties.

2.4 *Méthode de quasi-Newton (BFGS et LBFGS)*

2.4.1 *Idée générale*

L'idée des méthodes de quasi-Newton[2] est de remplacer l'inverse de la matrice Hessienne $[\nabla^2 f(x_k)]^{-1}$ par une estimée B_k^{-1} dépendant des précédents gradients $\nabla f(x_j)$ (qu'on a de toute façon déjà calculés). Intuitivement, il y a quelque chose à tirer de ces gradients : obtenus en des points différents, ils donnent une idée de la variation de la dérivée, et ainsi contraignent indirectement la matrice des dérivées secondes.

2. Voir chapitre 6 de Nocedal et Wright pour une explication détaillée des méthodes de quasi-Newton, avec les algorithmes DFP, BFGS, et SR1.

Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999. URL https://www.math.uci.edu/~qnie/Publications/NumericalOptimization.pdf

2.4.2 *Équation sécante*

Imaginons que la fonction à optimiser soit quadratique et ainsi avec une matrice Hessienne constante $\nabla^2 f$ qu'on suppose connue. Sous cette hypothèse, le gradient serait une fonction linéaire et la différence de deux gradient serait donnée directement par

$$\nabla f(x_{k+1}) - \nabla f(x_k) = \nabla^2 f \cdot (x_{k+1} - x_k) \tag{2.10}$$

Dans notre cas, on ne connaît pas la matrice Hessienne, et on ne suppose pas la fonction quadratique, mais il est raisonnable de demander que notre estimée B_{k+1} vérifie la même équation, dite *équation sécante*

$$\nabla f(x_{k+1}) - \nabla f(x_k) = B_{k+1} \cdot (x_{k+1} - x_k)\,. \tag{2.11}$$

Comme on voudra *in fine* plutôt l'inverse B_{k+1}^{-1} pour calculer la direction de descente, on peut multiplier par l'inverse à gauche et obtenir

$$B_{k+1}^{-1}\,[\nabla f(x_{k+1}) - \nabla f(x_k)] = x_{k+1} - x_k\,. \tag{2.12}$$

2.4.3 *Minimiser la distance à l'estimée précédente*

L'équation (2.12) est un système de n équations scalaires, mais B_{k+1}^{-1} a $n(n+1)/2$ composantes, et on a donc trop de liberté ! Pour fixer B_{k+1}^{-1} entièrement, on lui demande d'être le plus près possible de l'estimée précédente B_k et on résout ainsi un problème de la forme

$$B_{k+1}^{-1} = \underset{\substack{H=H^T \\ H[\nabla f(x_{k+1})-\nabla f(x_k)]=x_{k+1}-x_k}}{\operatorname{argmin}} \|H - B_k^{-1}\|_W^2 \tag{2.13}$$

où $\|A\|_W^2 = \operatorname{tr}(WAWA^T)$ définit une norme au carré pour une matrice de pondération $W \succ 0$ *a priori* arbitraire. Comment fixer cet ultime arbitraire dans le choix de la norme ? *Une* option[3] est de demander que W se comporte comme la matrice Hessienne, *i.e.* vérifie l'équation sécante $\nabla f(x_{k+1}) - \nabla f(x_k) = W \cdot (x_{k+1} - x_k)$. Cela ne contraint pas totalement la norme, mais fixe entièrement le minimum du problème d'optimisation (2.13) qui peut être résolu exactement pour fournir

$$B_{k+1}^{-1} = \left(I - \frac{\Delta x_k \Delta f_k^T}{\Delta x_k^T \Delta f_k}\right) B_k^{-1} \left(I - \frac{\Delta f_k \Delta x_k^T}{\Delta f_k^T \Delta x_k}\right) + \frac{\Delta x_k \Delta x_k^T}{\Delta x_k^T \Delta f_k}\,, \tag{2.14}$$

avec $\Delta x_k = x_{k+1} - x_k$ et $\Delta f_k = f(x_{k+1}) - f(x_k)$. On peut ainsi construire B_k itérativement, en partant de $B_0 = I$ (qui est encore un choix arbitraire). Utiliser la direction de descente $p_k = -B_k^{-1}\nabla f(x_k)$, idéalement avec un pas ℓ_k vérifiant les conditions de Wolfe, donne l'algorithme de Broyden-Fletcher-Goldfarb-Shanno, aka BFGS. À part l'équation sécante, le reste de la dérivation est arbitraire et d'autres choix donnent d'autres algorithmes de quasi-Newton. En pratique, BFGS est celui qui se comporte le mieux. On peut de plus montrer qu'il converge super-linéairement.

3. L'intérêt de ce choix est notamment de rendre le problème d'optimisation (2.13) soluble.

2.4.4 *L-BFGS*

L'algorithme BFGS demande toujours de stocker une matrice B_k^{-1} de taille $n \times n$ à chaque itération, avec un coût mémoire $\propto n^2$. Pour des problèmes de très grande taille, ce coût mémoire peut-être prohibitif. On peut alors ne stocker qu'une approximation de bas rang de B_k^{-1} en remarquant que l'équation (2.13) donne B_{k+1}^{-1}

en fonction de B_k^{-1} à partir de multiplications par l'identité et des matrices de rang 1. Ainsi, pour k pas trop grand, on peut construire l'action de B_k^{-1} sur un vecteur récursivement, en utilisant x_k, $f(x_k)$, et $\nabla f(x_k)$, sans jamais écrire explicitement B_k^{-1}. Si k est plus grand que n, cela n'est pas plus rapide que d'écrire B_k^{-1} car on stocke $k \geq n$ vecteurs de taille n. L'idée de l'algorithme L-BFGS (pour *limited-memory* BFGS) est d'introduire un paramètre de mémoire $m \ll n$. On construit à chaque étape $B_k^{-1} \cdot \nabla f(x_k)$ récursivement à partir des m points précédents comme si on avait démarré à l'itération $k-m$, de la matrice $B_{k-m}^{-1} = I$. Intuitivement, on ne perd pas grand chose à oublier les points éloignés de la valeur actuelle, où la matrice Hessienne était de toute façon probablement très différente. En pratique, des valeurs modestes de m peuvent fournir une performance comparable[4] à BFGS « standard » pour un coût mémoire $\propto n \times m \ll n^2$.

4. On peut en général utiliser $m \simeq 10$, et n'utiliser des valeurs plus grandes que si la matrice Hessienne semble être horriblement mal conditionnée.

2.5 *Méthode du gradient conjugué*

L'algorithme du gradient conjugué a une philosophie assez différente des méthodes de Newton et quasi-Newton, mais *in fine*, pour une fonction générique, il peut être vu comme une ultime simplification de L-BFGS avec $m = 1$, c'est à dire où on ne retient que l'itération précédente.

2.5.1 *Fonctions quadratiques*

L'idée du gradient conjugué vient comme pour Newton de la minimisation fonctions quadratiques $f : x \mapsto \frac{1}{2}x^T Q x - bx$ avec $Q \succ 0$. Cette fois-ci, on ne cherche pas à minimiser f en un coup (ce qui demande une inversion, prohibitive pour n grand) mais on se contente d'une minimisation en exactement n itérations où chaque itération est rapide.

L'intuition du gradient conjugué est que minimiser à fond (avec un pas optimal) successivement dans des directions « orthogonales » (pour la bonne notion d'orthogonalité) est suffisant pour minimiser une fonction quadratique. La bonne notion d'orthogonalité est associée à Q, *i.e.* au produit scalaire $\langle x, y \rangle_Q := x^T Q y$. Des vecteurs $p_1, \cdots p_l$ orthogonaux vis à vis de ce produit scalaire sont dits « Q-conjugués ». Avec ce produit scalaire, on a $f(x) = \|x\|_Q^2 - bx$. Ainsi, dans une base orthonormée pour ce produit scalaire (n vecteurs Q conjugués et normés), les lignes de niveau de f ne sont plus des ellipsoïdes mais des sphères (pour $n = 2$ des cercles). Minimiser à fond *successivement* dans chaque direction de la base permet clairement d'atteindre le minimum (ce qui ne serait pas le cas dans une base pour le produit scalaire usuel).

Cette idée peut être formalisée dans le théorème suivant

Théorème 22 (Convergence d'une descente selon des directions conjugées pour une fonction quadratique)**.** *On suppose qu'à chaque*

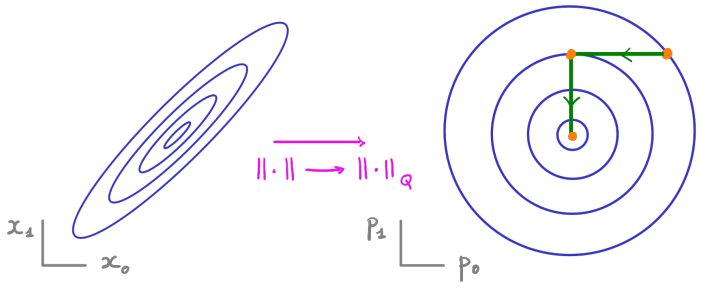


FIGURE 2.9: Intuition pour le gradient conjugué : en minimisant successivement selon n (ici 2) directions conjuguées (orthogonales pour le produit scalaire $\langle x, y\rangle_Q := x^T Q y$), on atteint le minimum. La même idée ne fonctionne pas pour le produit scalaire usuel : on oscillerait sans atteindre exactement le minimum.

itération

$$x_{k+1} = \ell_k p_k \tag{2.15}$$

où $p_1, p_2, \cdots, p_n$ sont des directions Q-conjuguées, et ℓ_k est le pas optimal à chaque étape donné par

$$\ell_k = -\frac{\nabla f(x_k)^T p_k}{p_k^T Q p_k} \tag{2.16}$$

Alors, le gradient conjugué atteint le minimum de f en au plus n itérations.

On pourrait trouver des directions conjuguées en diagonalisant Q, mais alors autant inverser Q et utiliser directement Newton, ce qui serait moins cher ! On va plutôt partir de la descente de gradient, et modifier la direction de descente à chaque étape pour garantir qu'elle soit conjuguée aux directions précédentes. On pose

$$p_k = -\nabla f(x_k) + \beta_k p_{k-1}\,, \tag{2.17}$$

la direction de descente donnée par l'opposé du gradient, et un terme proportionnel à la direction précédente. On peut montrer qu'en choisissant

$$\beta_k = \frac{\nabla f(x_k)^T Q p_{k-1}}{p_{k-1}^T Q p_{k-1}}, \tag{2.18}$$

les directions $p_0, \cdots, p_n$ sont bien Q conjuguées ! En utilisant cette propriété, on peut exprimer β_k sans faire référence à Q : $\beta_k = \frac{\|\nabla f(x_k)\|^2}{\|\nabla f(x_{k-1})\|^2}$. Utiliser cette direction de descente (2.17) avec le pas optimal $\ell_k = -\frac{\nabla f(x_k)^T p_k}{p_k^T Q p_k}$ donne l'algorithme du gradient conjugué pour une fonction quadratique, qui converge exactement vers le minimum en au au plus n étapes (et peut être très proche du minimum en beaucoup moins). On a ainsi obtenu une sorte d'intermédiaire entre la descente de gradient et Newton pour la minimisation de fonctions quadratiques (voir Fig. 2.10).

L'algorithme du gradient conjugué est une puissante méthode pour résoudre itérativement un système linéaire $Qx = b$ (dont la solution est minimum de $x^T Q x/2 - bx$) de très grande taille ou obtenir une approximation rapide de la solution. L'algorithme est en particulier utile pour des matrices peu denses (*sparse*), correspondant à des applications linéaires dont on sait écrire facilement l'action sur un vecteur, ce qui est la seule chose que demande l'algorithme.

2.5.2 *Fonction générique*

Pour une fonction générique, comment faire ? On n'a plus Q, qui correspondrait à la matrice Hessienne, et ainsi on ne connaît pas ℓ_k. On peut simplement croiser les doigts et utiliser la même direction de descente (2.17) avec un pas obtenu par une recherche linéaire

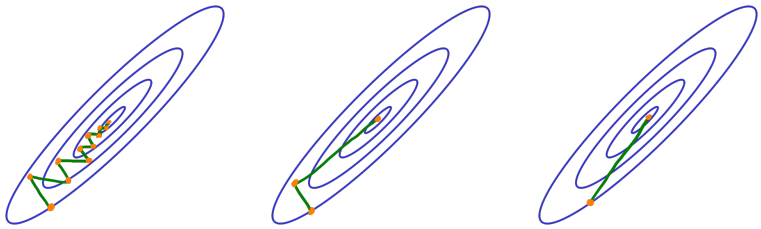


FIGURE 2.10: Convergence de la descente de gradient (gauche), du gradient conjugué (centre), et de la méthode de Newton (droite), à pas optimal et pour une fonction quadratique $f : x \in \mathbb{R}^n \mapsto \frac{1}{2}x^T Q x - bx$. La descente de gradient converge en un nombre infini d'itérations, le gradient conjugué en exactement n itérations (ici $n = 2$), et la méthode de Newton en 1 itération.

et vérifiant les conditions de Wolfe. C'est l'algorithme du gradient conjugué non-linéaire de Fletcher-Reeves, qui est la généralisation au fonctions quelconques la plus immédiate [5] du gradient conjugué

$$x_{k+1} = x_k + \ell_k p_k \tag{2.19}$$
$$p_k = -\nabla f(x_k) + \beta_k p_{k-1} \tag{2.20}$$
$$\beta_k = \frac{\|\nabla f(x_k)\|^2}{\|\nabla f(x_{k-1})\|^2} \tag{2.21}$$
$$\ell_k \quad \text{vérifiant Wolfe} \tag{2.22}$$

Évidemment, pour une fonction générique, le gradient conjugué ne converge plus en n itérations, mais on observe en pratique qu'il converge beaucoup plus vite que la descente de gradient usuelle, tout en étant extrêmement simple à implémenter. En particulier, on observe que le gradient conjugué oscille moins lors de la minimisation de fonctions à lignes de niveaux aplaties.

Pour une fonction générique, on peut se donner une intuition à l'excellente performance du gradient conjugué en rapprochant l'algorithme des méthodes de *momentum* [6]. Imaginons que l'on est dans un cas difficile, typiquement avec beaucoup d'oscillations, où la norme du gradient est faible mais varie peu d'une itération à l'autre. Alors la direction de descente donnée par le gradient conjugué est $p_k \simeq p_{k-1} - \nabla f(x_k)$ où le second terme est très petit. Le gradient ne fixe ainsi plus la direction de descente mais sa dérivée $p_k - p_{k-1}$. Intuitivement, cela revient à donner de l'inertie à la direction de descente, qui réduit ainsi les oscillations et aide à dévaler la pente dans rebondir de chaque côté de la vallée.

5. On peut choisir d'autres formules pour β_k qui coïncident pour les fonctions quadratiques, et donnent dans ce cas les mêmes directions conjuguées, mais différent pour une fonction générique. Par exemple, Polak et Ribière ont proposé

$$\beta_k = \frac{\|\nabla f(x_k)\|^2 - \nabla f(x_k)^T \nabla f(x_{k-1})}{\|\nabla f(x_{k-1})\|^2}$$

qui peut converger un peu plus vite en pratique que (2.21).

6. Voir par exemple cet article de Goh *Why momentum really works* pour une explication de l'efficacité du momentum en optimisation.

2.5.3 *Résumé*

On a vu de nombreuses manières de choisir la direction de descente, ce qui amène naturellement la question :

Que faut il choisir en pratique ?

Comme souvent pour ce type de questions, la réponse est

Ça dépend !

Clairement, la descente de gradient n'a pas d'avantage si ce n'est une extrême simplicité, et on privilégie en général les autres algorithmes. Ensuite, il est plus difficile de choisir (voir Table 2.1) en particulier entre BFGS et gradient conjugué. En pratique, le gradient conjugué converge un peu plus lentement en *fonction du*

nombre d'itérations que BFGS ou L-BFGS avec $m \simeq 10$, mais le coût ajouté par itération est en contrepartie plus faible. Ainsi, on peut privilégier le gradient conjugué si la fonction et son gradient sont extrêmement rapides à évaluer (quelques opérations arithmétiques), et au contraire choisir BFGS ou LBFGS avec m grand si le coût d'évaluation de la fonction et de son gradient domine[7] devant les opérations que l'on effectue ensuite dessus pour trouver la direction de descente. Enfin, Newton a un coût très important par itération mais est robuste à un extrême aplatissement des lignes de niveau. On privilégie cet algorithme pour les fonctions avec peu de variables mais pour lesquelles on observe une convergence très lente des autres méthodes.

7. Un autre intérêt de BFGS et L-BFGS est que le pas $\ell_k \equiv 1$ vérifie quasi systématiquement les conditions de Wolfe (au moins après quelques itérations), ce qui évite une recherche linéaire pour la plupart des itérations et ainsi réduit le nombre d'évaluations de la fonction et de son gradient.

	Direction	Efficacité	Mémoire	Calcul
Gradient	$-\nabla f(x_k)$	♡	$O(n)$	$O(n)$
Gradient conjugué	$-\nabla f(x_k) + \beta_{k-1} p_{k-1}$	♡♡	$O(n)$	$O(n)$
L-BFGS (mémoire m)	$-\tilde{B}_k^{-1} \nabla f(x_k)$	♡♡ $\underset{m \to +\infty}{\longrightarrow}$ ♡♡♡	$O(nm)$	$O(nm)$
BFGS	$-B_k^{-1} \nabla f(x_k)$	♡♡♡	$O(n^2)$	$O(n^2)$
Newton	$-\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$	♡♡♡♡	$O(n^2)$	$O(n^3)$

TABLE 2.1: Résumé des principales directions de descente et leurs caractéristiques. Pour les coûts mémoire et calcul, on néglige les coûts d'évaluation de la fonction et de son gradient (ou de manière équivalente, on suppose qu'ils sont au plus $O(n)$).

3
Généralités théoriques pour l'optimisation sous contraintes

3.1 *Contraintes d'égalité*

On cherche ici à minimiser f sur ses variables en vérifiant en même temps n contraintes d'égalité

$$\begin{aligned} &\min_z f(z) \\ &\text{tel que } \; c(z) = 0\,. \end{aligned} \tag{3.1}$$

Ici $z \in \mathbb{R}^N$ et $c : \mathbb{R}^N \to \mathbb{R}^n$. En supposant qu'on n'a pas de contraintes redondantes, on a ainsi n contraintes scalaires qui réduisent d'autant le nombre de « vrais » degrés de libertés.

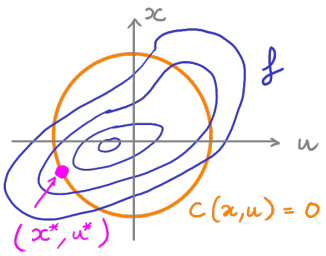


FIGURE 3.1: Minimisation d'une fonction $f : \mathbb{R}^2 \to \mathbb{R}$ avec une contrainte d'égalité $c(x,u) = x^2 + y^2 - 1 = 0$. La contrainte est localement inversible, par exemple $x(u) = \sqrt{1-u^2}$ pour $x \geq 0$, mais pas globalement (x n'est pas une fonction de u).

3.1.1 *Élimination des variables*

Une première approche est de résoudre la contrainte et de séparer z en $u \in \mathbb{R}^m$ et $x \in \mathbb{R}^n$ avec $m := N - m$. Le vecteur u correspond aux degrés de libertés restants, non contraints, et on suppose qu'on peut au moins localement exprimer x en fonction de u grâce à l'équation de contrainte $c(x,u) = 0$ (voir Fig. 3.1).

Avec cette notation, on peut regarder comment varie f en fonction de u *quand x est asservi à u pour vérifier la contrainte, i.e.* $x = x(u)$. On a :

$$\frac{\mathrm{d}f}{\mathrm{d}u} = \frac{\partial f}{\partial u} + \frac{\partial f}{\partial x}\frac{\partial x}{\partial u} \tag{3.2}$$

où la dérivée droite dénote une dérivée totale par rapport à u. En dérivant la contrainte $c(x(u),u) = 0$ on a

$$\frac{\mathrm{d}c}{\mathrm{d}u} = \frac{\partial c}{\partial u} + \frac{\partial c}{\partial x}\frac{\partial x}{\partial u} = 0 \tag{3.4}$$

ce qui implique

$$\frac{\partial x}{\partial u} = -\left[\frac{\partial c}{\partial x}\right]^{-1}\frac{\partial c}{\partial u} \tag{3.5}$$

et enfin

$$\frac{\mathrm{d}f}{\mathrm{d}u} = \frac{\partial f}{\partial u} - \frac{\partial f}{\partial x}\left[\frac{\partial c}{\partial x}\right]^{-1}\frac{\partial c}{\partial u} \tag{3.6}$$

Un point *stationnaire* $z^* = (x^*, u^*)$ est un point où $\frac{\mathrm{d}f}{\mathrm{d}u}$ s'annule (et en général $\frac{\partial f}{\partial u}(x^*,u^*) \neq 0$).

Attention à la notation compacte utilisée ici. Les vecteurs $\frac{\mathrm{d}f}{\mathrm{d}u}$ et $\frac{\partial f}{\partial u}$ appartiennent à $\mathbb{R}^m$, $\frac{\mathrm{d}f}{\mathrm{d}u}$ à $\mathbb{R}^n$ et $\frac{\partial x}{\partial u}$ est une matrice avec m lignes et n colonnes. En composantes, l'équation (3.2) donne par exemple :

$$\frac{\mathrm{d}f}{\mathrm{d}u_i} = \frac{\partial f}{\partial u_i} + \sum_{j=1}^{n}\frac{\partial f}{\partial x_j}\frac{\partial x_j}{\partial u_i}\,. \tag{3.3}$$

3.1.2 *Équations adjointes – multiplicateurs de Lagrange*

On ajoute maintenant n *multiplicateurs de Lagrange* regroupés en un vecteur $\lambda \in \mathbb{R}^n$ et on regarde le lagrangien $\mathcal{L} : \mathbb{R}^{2n+m} \to \mathbb{R}$

$$\mathcal{L}(z, \lambda) := f(z) + \lambda^T c(z) \tag{3.7}$$

Trouver un point stationnaire de $\mathcal{L}$ donne un point stationnaire de f sous la contrainte, *i.e.* formellement :

Théorème 23. *z^* est un point stationnaire de f sous la contrainte $\Leftrightarrow \exists \lambda^* \in \mathbb{R}^n$ tel que (z^*, λ^*) est un point stationnaire de $\mathcal{L}$*

Idée de preuve. $\Leftarrow$ Être un point stationnaire de $\mathcal{L}$ implique :

$$0 = \frac{\partial \mathcal{L}}{\partial \lambda} = c(z^*) \quad \textit{i.e.} \quad \text{la contrainte est vérifiée} \tag{3.8}$$

$$0 = \frac{\partial \mathcal{L}}{\partial z} = \frac{\partial f}{\partial z} + \lambda^{*T} \frac{\partial c}{\partial z} \tag{3.9}$$

La deuxième équation dit que le gradient est une combinaison linéaire des vecteurs normaux à chaque composante de la contrainte (voir Fig. 3.2). Ainsi une petite variation de f dans la direction δz *vérifiant la contrainte* donne

$$\begin{aligned} f(z + \delta z) - f(z) &= \frac{\partial f}{\partial z} \delta z + O(\|\delta z\|^2) \\ &= -\lambda^{*T} \frac{\partial c}{\partial z} + O(\|\delta z\|^2) \\ &= 0 + O(\|\delta z\|^2) \end{aligned} \tag{3.10}$$

donc point stationnaire de f sous la contrainte.

$\Rightarrow$ On sépare de nouveau formellement z en (x, u) avec localement x fonction de u. On pose

$$\lambda^* = \frac{\partial f}{\partial x} \left[\frac{\partial c}{\partial x} \right]^{-1} \tag{3.11}$$

Que $z^* = (x^*, u^*)$ soit un point fixe de f sous la contrainte implique que z^* vérifie la contrainte, donc $c(z^*) = 0$ donc $\frac{\partial \mathcal{L}}{\partial \lambda} = 0$. Ensuite on a

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial f}{\partial x} - \frac{\partial f}{\partial x} \left[\frac{\partial c}{\partial x} \right]^{-1} \frac{\partial c}{\partial x} = 0 \quad \text{trivialement} \tag{3.12}$$

$$\frac{\partial \mathcal{L}}{\partial u} = \frac{\partial f}{\partial u} - \frac{\partial f}{\partial x} \left[\frac{\partial c}{\partial x} \right]^{-1} \frac{\partial c}{\partial u} = 0 \quad \text{par stationnarité – eq. (3.6)} \tag{3.13}$$

donc point stationnaire de $\mathcal{L}$.

En *bonus*, l'approche lagrangienne nous fournit la sensibilité du minimum trouvé à une modification de la contrainte (voir Fig. 3.3).

Théorème 24. *Si (z^*, λ^*) est un point stationnaire de $\mathcal{L}$*

$$\frac{\partial f}{\partial c}(z^*) = \lambda^* \tag{3.14}$$

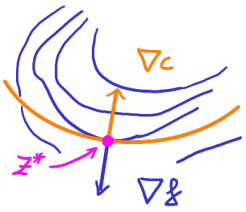


FIGURE 3.2: En un point stationnaire, le gradient de f est une combinaison linéaire du gradient de chaque composante de la contrainte (ici il n'y a qu'une contrainte). Si ça n'était pas le cas, on pourrait faire varier f localement au premier ordre sans briser la contrainte, et on ne serait donc pas sur un point stationnaire.

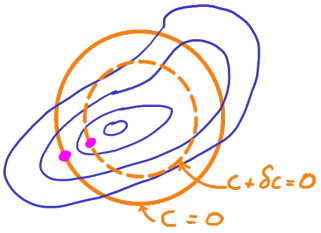


FIGURE 3.3: Si on modifie infinitésimalement la contrainte $c(x, y) = x^2 + y^2 - 1 \to x^2 + y^2 - 1 + \delta c$, le minimum sous la contrainte se déplace, et la variation correspondante de $f(z^*)$ vaut $\lambda\, \delta c$.

3.2 *Contraintes d'inégalités*

On cherche ici à minimiser f sur ses variables en vérifiant en même temps n contraintes d'inégalité [1]

$$\begin{aligned} &\min_x f(x) \\ &\text{tel que } \; c(x) \leq 0\,. \end{aligned} \tag{3.15}$$

1. Le sens de l'inégalité « $\leq$ » est *a priori* arbitraire mais il faut s'y tenir pour la suite, sans quoi de nombreux signes sont opposés.

Minimiser avec une contrainte d'inégalité se ramène en pratique à optimiser avec soit

— une contrainte d'égalité – « contrainte active »

— pas de contrainte – « contrainte inactive » ,

voir Fig. 3.4 pour une illustration. Ainsi, il n'y a pas de lien entre

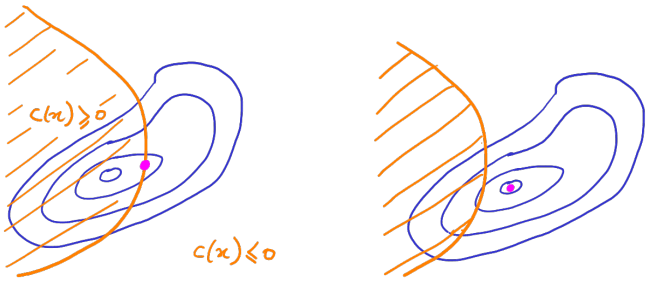


FIGURE 3.4: Une contrainte d'inégalité est soit *active*, le minimum étant au bord [la contrainte est alors saturée $c(x^*) = 0$] soit *inactive*, auquel cas le minimum est au même endroit que sans contrainte, $\nabla f(x^*) = 0$ et $c(x^*) < 0$.

le nombre de degrés de libertés effectifs et le nombre d'inégalités (comme c'était le cas pour le cas d'égalité), car toutes les contraintes ne sont pas a priori active (ni activables) en même temps (voir Fig. 3.5).

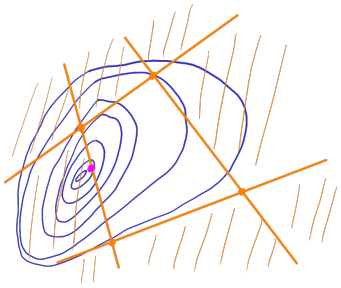


FIGURE 3.5: Avec plusieurs contraintes d'inégalités régulières (ici linéaires, on peut obtenir un ensemble d'optimisation non régulier, avec des « coins » correspondant à l'intersection de plusieurs contraintes. Il n'y a pas de lien a priori entre le nombre de contraintes et le nombre de degrés de libertés effectifs. Par exemple ici, deux contraintes au maximum sont actives en même temps.

3.2.1 *Contraintes actives ou inactives*

Évidemment, on ne sait pas a priori dans quel cas on va se trouver. Les conditions de Karush-Kuhn-Tucker (KKT) permettent de garantir qu'on a bien minimisé f avec les bonnes contraintes actives (on ne s'est pas mis au bord du domaine alors qu'on n'avait pas besoin).

Théorème 27 (Conditions KKT). *$x^* \in \mathbb{R}^n$ solution du problème (3.15) avec m contraintes d'inégaltiés [et, condition technique, $\{\nabla c_i(x^*), i = 1 \dots m\}$ est une famille libre]* $\implies$

$$\nabla f(x^*) = -\sum_{i=1}^{m} \lambda_i \nabla c_i(x^*) \tag{3.16}$$

$$\lambda_i \geq 0 \text{ pour } i = 1, \dots, m \tag{3.17}$$

$$\lambda_i c_i(x^*) = 0 \text{ pour } i = 1, \dots, m \tag{3.18}$$

La condition technique est aussi appelée qualification des contraintes, ou régularité, et permet d'éliminer le cas où l'optimum est pile sur un point singulier (par exemple de rebroussement) de la contrainte.

Les trois conditions sont nécessaires (voir Fig. 3.6. La première dit que le gradient est combinaison linéaire des vecteurs orthogonaux aux contraintes actives. La deuxième dit qu'on est sur le bord si minimiser davantage nous ferait violer la contrainte (elle évite

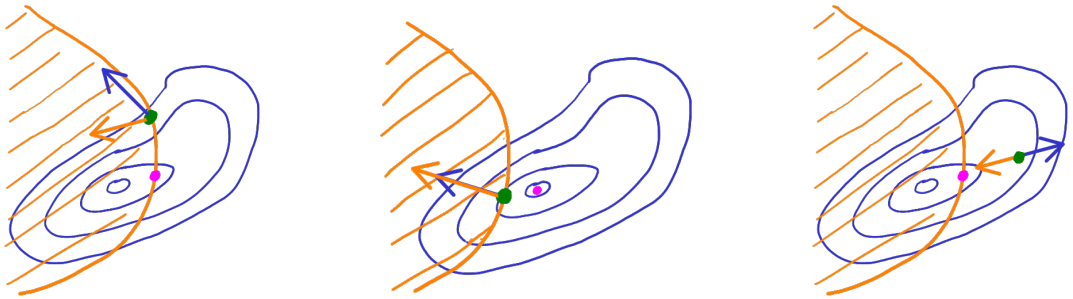


FIGURE 3.6: En violet, le vrai minimum sous contrainte, en vert, un candidat qui vérifie toutes les conditions KKT sauf la première (à gauche), la deuxième (au milieu), la troisième (à droite), illustrant la nécessité des trois conditions.

qu'on soit sur le bord alors que ça n'était pas nécessaire). La troisième dit qu'on est soit sur le bord ($c_i = 0$), soit à l'intérieur auquel cas le gradient de f n'a pas de raison de mettre en jeu la contrainte ($\lambda_i = 0$).

En pratique, une approche pour minimiser une fonction avec des contraintes d'inégalité *en petit nombre* est de choisir une combinaison des contraintes, de supposer qu'elle sont actives, puis de minimiser, et vérifier KKT sur le minimum trouvé. Si KKT n'est pas vérifié, on essaie un autre ensemble de contraintes actives et on recommence. En général, on voit que c'est une méthode peu efficace s'il y a beaucoup de contraintes, car pour n contraintes, on a 2^n combinaisons possibles de contraintes actives.

4
Minimisation de fonctions non-différentiables

4.1 Introduction

4.1.1 Pourquoi c'est utile

Les fonctions non-différentiables apparaissent par exemple quand on considère une fonction f définie comme le maximum de plusieurs fonctions g_i (qui peuvent être régulières). Dans ce cas, on est typiquement non-différentiable lorsque l'on change de fonction $g_i \to g_j$ qui est maximum.

Les fonctions non-différentiables apparaissent aussi naturellement quand on veut écrire un problème de minimisation sous contraintes d'égalités comme une minimisation sans contrainte mais avec une *pénalisation* :

$$\min_{x \in \mathbb{R}^n} f(x) + \gamma \|c(x)\|_1 \tag{4.1}$$

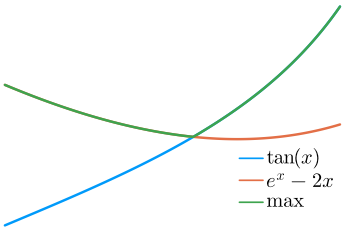


FIGURE 4.1: Le maximum de deux fonctions est en général non-différentiable, ici précisément au minimum.

Naïvement, on retrouve le problème d'optimisation sous contrainte quand $\gamma \to +\infty$. En réalité, lorsque γ est suffisamment grand, disons plus grand que la constante de Lipschitz de f, il devient favorable de minimiser *exactement* le terme de pénalisation de la contrainte (et pas juste approximativement) et la pénalisation est ainsi *exacte* sans prendre $\gamma \to +\infty$. La non-différentiabilité au voisinage de l'annulation de la contrainte permet que la pente du terme de pénalité « gagne » systématiquement contre le gradient de f au voisinage de son minimum.

Ou la régularisation Lasso (Least Absolute Shrinkage and Selection Operator) d'une régression linéaire

$$\operatorname*{argmin}_x \|Ax - b\|_2^2 + \gamma \|x\|_1 \tag{4.2}$$

4.1.2 Pourquoi ça n'est pas évident

4.2 Régularisation

Pour une fonction convexe f, et $\mu > 0$, on définit *l'enveloppe de Moreau* de f

$$M_{f,\mu} = \min_{s \in \mathbb{R}^n} \left[f(s) + \frac{1}{2\mu} \|x - s\|^2 \right] \tag{4.3}$$

Intuitivement, quand μ est très petit, le terme quadratique domine, et la minimisation force $s \simeq x$ et ainsi $M_{f,\mu}(x) \simeq f(x)$. Pour comprendre pourquoi cette transformation régularise f, on peut regarder l'exemple de la valeur absolue :

$$f(x) = |x| \tag{4.4}$$

$$M_{f,\mu}(x) = \begin{cases} |x| - \frac{\mu}{2} & \text{si } |x| \geq \mu \\ \frac{x^2}{2\mu} & \text{si } |x| < \mu \end{cases} \tag{4.5}$$

où on voit que $M_{f,\mu}$ est bien une fonction lisse, qui s'approche de la fonction originelle quand $\mu \to 0$. On note aussi que le minimiseur ($x^* = 0$) n'est pas changé par la régularisation. Il s'agit en fait de deux propriétés générales de l'enveloppe de Moreau :

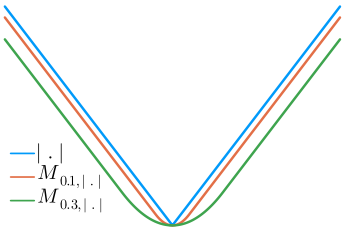


FIGURE 4.2: Valeur absolue et son enveloppe de Moreau pour $\mu = 0.1$ et $\mu = 0.3$.

— $M_{f,\mu}$ est de classe C^1, donc bien « régulière »

— Un minimiseur global de f est minimiseur global de $M_{f,\mu}$ et réciproquement

Ainsi, avec l'enveloppe de Moreau, on a régularisé f sans modifier son minimiseur ! Le prix à payer est évidemment que $M_{f,\mu}$ est donnée par un nouveau problème de minimisation. Mais au moins théoriquement, on s'est ramené au cas différentiable, qu'on sait gérer par exemple avec des algorithmes de descente.

Pour minimiser $M_{f,\mu}$ et donc *in fine* f, il est naturel calculer son gradient. Celui-ci met en jeu l'opérateur proximal, qui donne le point $s \in \mathbb{R}^n$ auquel le minimum de (4.3) est atteint :

$$\text{Prox}_{\mu,f}(x) = \underset{s \in \mathbb{R}^n}{\text{argmin}} \left[f(s) + \frac{1}{2\mu} \|x - s\|^2 \right] \tag{4.6}$$

L'opérateur proximal interpole entre le minimum x^* de f et la distance à x (quand $\mu \to 0$, $\text{Prox}_{\mu,f}(x) = x$ et quand $\mu \to +\infty$ $\text{Prox}_{\mu,f}(x) = x^*$). On peut exprimer le gradient de l'enveloppe de Moreau en fonction du l'opérateur proximal

Théorème 34. *f convexe, $\mu > 0 \implies$ f est $\mathcal{C}^1$ et :*

$$\nabla M_{\mu,f}(x) = \frac{1}{\mu} \left[x - \text{Prox}_{\mu,f}(x) \right] \tag{4.7}$$

Le minimum de x^* de f est aussi minimum de l'enveloppe de Moreau, et donc son gradient s'annule, et ainsi on a $\text{Prox}_{\mu,f}(x^*) = x^*$. Autrement dit, le minimum est un point fixe de l'opérateur proximal.

4.3 *Conjugaison convexe*

Un outil utile, particulièrement dans le contexte non-différentiable, est la transformée de Fenchel (ou conjuguée convexe) de f :

$$f^*(\varphi) = \sup_{x \in \mathbb{R}^n} \left[\varphi^T x - f(x) \right] . \tag{4.8}$$

C'est une fonction convexe. En itérant deux fois la transformée, on retombe sur f si et seulement si f est convexe. Sinon la « biconjuguée » f^{**} est la fonction dont l'épigraphe est l'enveloppe convexe

de celui de f (essentiellement, f^{**} est la fonction convexe la plus proche de f si f ne l'est pas).

La transformée de Fenchel se comporte comme un inverse pour la pente de f. Pour une fonction convexe on a en effet le lemme suivant

$$\phi \in \partial f(x) \Leftrightarrow x \in \partial f^*(\phi) \tag{4.9}$$

Suivant les cas, la conjuguée convexe f^* peut être plus simple que f, ce que l'on peut exploiter par exemple pour calculer plus facilement un opérateur proximal en utilisant l'identité de Moreau

Théorème 36 (Identité de Moreau). *Soit $f : \mathbb{R}^n \to \mathbb{R}$ convexe*

$$x = \mathrm{Prox}_{\mu,f}(x) + \mu \,\mathrm{Prox}_{1/\mu,f^*}(x/\mu) \tag{4.10}$$

4.4 Algorithmes

4.4.1 Méthode naïve

Pour minimiser une fonction non-différentiable, on peut oublier toute la théorie que l'on vient de voir, fermer les yeux, et faire une descente de gradient en utilisant simplement *un* sous-gradient de la fonction.

$$x_{k+1} = x_k - \ell_k g_k \quad \text{avec} \quad g_k \in \partial f(x_k) \tag{4.11}$$

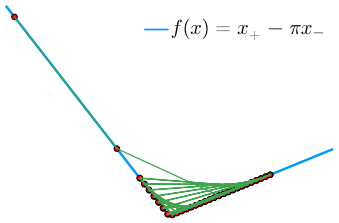


FIGURE 4.3: Descente avec de gradient pour la fonction $x \mapsto x_+ - \pi x_-$. On oscille au voisinage du minimum, sans l'atteindre (mais sans exploser).

Si le minimum est dans un coin, comme pour la valeur absolue, le gradient n'est pas petit près du minimum, et ainsi rien n'empêche les oscillations avec un pas $\ell_k = \ell$ fixe (même petit). Au moins, si le pas est assez petit, la descente est« stable », dans le sens que l'algorithme n'explose pas, et les oscillations autour du minimum sont d'autant plus faibles que le pas est petit. C'est l'objet du théorème suivant

Théorème 39 (Convergence de la descente de gradient à pas fixe). *Soit f convexe de sous-gradient borné par G localement,*

$$\exists k \in \mathbb{N},\ \|f(x_k) - f(x^*)\| \leq G\ell \tag{4.12}$$

On peut raffiner la constante en $G\ell(1+\varepsilon)/2$ pour tout $\varepsilon > 0$

On voit ainsi qu'en pratique, si on réduit suffisamment lentement le pas ℓ_k plutôt que de le garder constant, on doit s'approcher du minimum. Il ne faut pas le réduire trop vite, sinon on n'a pas le temps de converger. Typiquement, on demande que $\sum_k \ell_k \to +\infty$ avec $\ell_k \to 0$. Par exemple, $\ell_k \propto 1/k$. Sous cette condition on a

Théorème 40. *Soit f convexe, sous-gradient borné, $\sum_k \ell_k \to +\infty$, $\ell_k \to 0$, alors la descente de gradient à pas ℓ_k converge vers le minimum de f.*

Le problème de cette approche est évidemment que la convergence est extrêmement lente.

4.4.2 *Algorithme proximal*

En général, l'idéal serait de faire une descente de gradient pour l'enveloppe de Moreau, c'est à dire d'itérer

$$x_{k+1} = Prox_{\mu,f}(x_k)\,. \tag{4.13}$$

Évidemment, pour une fonction générique, cela demande de résoudre à chaque étape un problème de minimisation complet qui n'a aucune raison d'être plus simple.

En revanche, dans la plupart des situations pratiques, le problème de minimisation se décompose de la manière suivante :

$$\min_{x\in\mathbb{R}^n} f(x) \;\text{ avec }\; f(x) = g(x) + h(x) \tag{4.14}$$

où g est lisse mais potentiellement compliquée à minimiser, et où h est non-lisse, mais très simple (par exemple $\gamma\|\cdot\|_1$ comme dans la régularisation Lasso ou la pénalisation exacte d'une contrainte). Ainsi, même s'il est difficile de calculer $\mathrm{Prox}_{\mu,f}$ pour faire une descente proximale de l'ensemble, on a des chances de pouvoir calculer exactement $\mathrm{Prox}_{\mu,h}$. On peut alors itérer une succession d'une descente de gradient lisse pour g et une descente de l'enveloppe de Moreau de h ce qui donne l'algorithme

$$x_{k+1} = \mathrm{Prox}_{\ell,h}\left(x_k - \ell\nabla g(x_k)\right) \tag{4.15}$$

Le point fixe de cet algorithme de descente est bien le minimum de la fonction. En effet :

$$x^* = \mathrm{Prox}_{\ell,h}\left(x^* - \ell_k\nabla g(x^*)\right) \implies 0 \in \partial h(x^*) + \nabla g(x^*)\,. \tag{4.16}$$

De plus on peut montrer que si $\ell \leq 1/L$ où L est la constante de Lipschitz du gradient de f, alors l'algorithme (4.15) converge.

5
Bibliographie

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. URL https://web.stanford.edu/~boyd/cvxbook/.

Delphine Bresch-Pietri and Nicolas Petit. Optimisation, 2022.

Yurii Nesterov. *Introductory lectures on convex optimization : A basic course*, volume 87. Springer Science & Business Media, 2003.

Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999. URL https://www.math.uci.edu/~qnie/Publications/NumericalOptimization.pdf.