Optimisation Non Linéaire Sous Contraintes

A. Bazeniar

Enseignant chercheur
Centre universitaire Abdelhafid Boussouf
Mila, Algérie

Se reporter à des manuels de base et à certaines recherches

Septembre 2023

Chapitre 1

Méthodes de résolution

1.1 Introduction

Les méthodes de résolution de programmes mathématiques non linéaires s'est développée rapidement depuis 1955. Beaucoup de méthodes proposées pour la résolution des problèmes avec contraintes conduisent à remplacer le problème donné par une suite de problèmes sans contraintes, dont les résolutions utilisent des méthodes classiques. Dans le reste du chapitre on définit le problème d'optimisation comme,

$$\begin{cases} Min \quad f(x) \\ S.C. \;\; x \in S \end{cases}$$

Avec f une fonction convexe et S convexe et fermé.

1.2 Notions algorithmiques utiles

Définition 1.2.1. *Soit l'application vectoriel $M : \mathbb{R}^{\mathbb{N}} \to \mathbb{R}^{\mathbb{N}}$. $(x^n)_{n\in\mathbb{N}}$ est la suite générée par cette application comme suit,*

$$x^{n+1} = M(x^n), \quad x^0 \text{ est la phase initiale.} \tag{1.1}$$

Définition 1.2.2. *On dit qu'un algorithme est convergent s'il atteint son minimiseur x. Leur efficacité est mesurée par,*

- ***vitesse de convergence :*** *qui mesure la rapidité d'atteindre x cherché.*
- ***complexité :*** *qui mesure le coût par chaque itération afin d'obtenir la solution x souhaitée, et ça avec une precision ϵ fixée.*

Remarque 1.2.1.

1. *Le critère d'arrêt de l'algorithme peut être exprimé de la forme suivante,*

$$\| \nabla f(x^{(n)}) \| \leq \epsilon. \tag{1.2}$$

2. *La convergence est linéaire si,*

$$\exists C \in [0,1[, \forall x^{(0)} \in \mathbb{R}^{\mathbb{N}}, err^{(n+1)} \leq C\ err^{(n)}, \tag{1.3}$$

avec $err^{(n)} = \| x - x^{(n)} \|$, l'erreur à l'iteration n.

3. *La convergence est d'ordre p si elle satisfait la relation,*

$$\exists C \in [0,1[, \forall x^{(0)} \in \mathbb{R}^{\mathbb{N}}, err^{(n+1)} \leq C\ (err^{(n)})^p, \tag{1.4}$$

si $p = 2$ la vitesse est dite quadratique.

Algorithmes de descente. Les algorithmes que nous allons considérer pour les problèmes d'optimisation ont la forme générale suivante :

$$x^{(0)} \text{ étant donné, calculer } x^{(n+1)} = x^{(n)} + t_n d^{(n)}. \tag{1.5}$$

Le vecteur $d^{(n)}$ s'appelle la direction de descente, et le réel $t_k > 0$ le pas de la méthode à la n-ième itération. On pratique, on choisira la direction et le pas afin que l'inégalité suivante soit satisfaite :

$$f(x^{(n+1)}) \leq f(x^{(n)}).$$

De tels algorithmes sont appelés ***algorithmes de descente***.

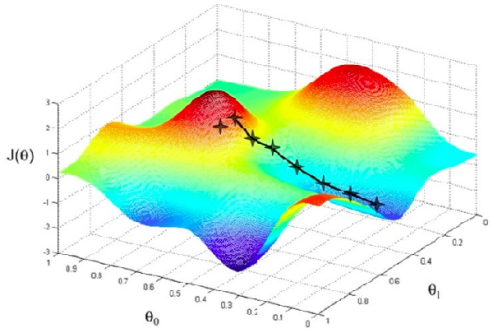

FIGURE 1.1 – Descente en gradient

Proposition 1.2.1. *(Interprétation géométrique du gradient)* $f : S \subset \mathbb{R}^{\mathbb{N}} \to \mathbb{R}^{\mathbb{N}}$ *de classe* C^1.

- *L'opposé du gradient,* $-\nabla f(x)$*, est une direction de descente en* x *(si* $\nabla f(x) \neq 0$*).*
- *La direction du gradient est la direction de plus forte pente (si* $\| d \| = 1$*, alors la pente dans la direction* d *correspond à la dérivée de* h *en* 0*).*
- *Le gradient est orthogonal à la tangente à la courbe en* x*.*

Algorithmes de gradient. Appliquant l'algorithme de descente décrit au dessus. La formule de Taylor en $x^{(n)}$ est,

$$f(x^{(n+1)}) = f(x^{(n)}) + < \nabla f(x^{(n)}), t_n d(n) > + o(t_n d^{(n)}).$$

Pour s'assurer que $f(x^{(n+1)}) - f(x^{(n)}) \leq 0$,
le choix de $d^{(n)}$ est donc,

$$d^{(n)} = -\nabla f(x^{(n)}). \tag{1.6}$$

Remarque 1.2.2. *Le pas* t_k *peut être choisi en tant que pas fix ou pas optimal (vitesse de convergence).*

1.3 Méthodes de Linéarisation

Parmi les méthodes de Linéarisation on trouve,

- Méthodes Tangentielle (centres linéarisées, Frank Wolf),
- Méthodes Barycentriques.

Dans cette section on choisit de développer la méthode de Frank Wolf.

1.3.1 Méthode de Frank Wolf (1956)

On applique la méthode pour la résolution des programmes à fonction objective quadratique et à contraintes linéaires.

1.3.1.1 Principe de la méthode

Le choix de la direction de descente consiste à linéariser en $x^{(k)}$ la fonction $f(x)$ en construisant une fonction linéaire tel que,

$$f(x^{(k+1)}) = f(x^{(k)}) + \nabla f(x^{(k)})^T (x^{(k+1)} - x^{(k)}). \tag{1.7}$$

Puis de minimiser cette fonction sur le domaine réalisable S.

1.3.1.2 Construction de la Méthode

- Soit $x^{(0)}$ un point arbitraire et $x^{(k)} \in S$. Pour construire $x^{(k+1)}$ on a,

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + t_k d^{(k)}, \\ &= x^{(k)} + t_k (y^{(k)} - x^{(k)}), \\ &= (1 - t_k) x^{(k)} + t_k y^{(k)}. \end{aligned}$$

  $x^{(k+1)}$ est en combinaison linéaire avec $x^{(k)}$ et $y^{(k)}$.
- Maintenant, soit le polynôme de Taylor de degré 1 autour de $x^{(k)}$,

$$\begin{aligned} f(y) &= f(x^{(k)}) + \nabla f(x^{(k)})^T (y - x^{(k)}), \\ &= f(x^{(k)}) + \nabla f(x^{(k)})^T y - \nabla f(x^{(k)})^T x^{(k)}). \end{aligned}$$

  Comme $x^{(k)}$ est fixe, alors minimiser $f(y)$ est équivalent à minimiser,

$$\nabla f(x^{(k)})^T y. \tag{1.8}$$

  Conséquence : $y^{(k)}$ est la solution optimale du problème P_k suivant,

$$\begin{cases} Min \;\; \nabla f(x^{(k)}) y \\ S.C. \;\; y \in S \end{cases}$$

  Question : $d^{(k)}$ est-elle une direction de descente ?

$$\begin{aligned} \nabla f(x^{(k)})^T d^{(k)} &= \nabla f(x^{(k)})^T (y^{(k)} - x^{(k)}), \\ &= \nabla f(x^{(k)})^T y^{(k)} - \nabla f(x^{(k)})^T x^{(k)}, \\ &< 0. \end{aligned}$$

  Car $y^{(k)}$ est améliorante.

- On détermine Le pas t_k^* qu'est une solution du problème,

$$f((1-t_k^*)x^{(k)} + t_k^* y^{(k)}) = \min_{0 \le t_k \le 1} \{f((1-t_k)x^{(k)} + t_k y^{(k)})\}. \tag{1.9}$$

- Le critère d'arrêt de la méthode est que $x^{(k)}$ soit solution du problème P_k i.e.

$$\nabla f(x^{(k)})^T (y^{(k)} - x^{(k)}) \ge 0. \tag{1.10}$$

1.3.1.3 Algorithme de Frank Wolfe

1. Soit $x^{(0)} \in S$; point arbitraire.
2. **Pour** $k = 0$ à N **faire**
   $r^{(k)} := \nabla f(x^{(k)})$.
   $y^{(k)} := \arg\min_{y \in S} < y, r^{(k)} >$.
   $g_k :=< x^{(k)} - y^{(k)}, r^{(k)} >$.
3. **Si** $g_k < \epsilon$ **alors** renvoyer $x^{(k)}$.
4. $t_k^* := \arg\min 0 \le t_k \le 1 f((1-t_k)x^{(k)} + t_k y^{(k)})$.
5. $x^{(k+1)} := (1-t_k^*)x^{(k)} + t_k * y^{(k)}$.
6. **Fin pour.**

Exemple 1.3.1. *Soit le problème suivant,*

$$\begin{cases} Min \;\; f(x) = 1/2(x_1^2 + x_2^2) \\ -x_1 + x_2 \le 7 \\ x_1 + x_2 \le 5 \\ -x_2 \le -2 \end{cases}$$

Avec $x^{(0)} = (-2, 3)$.
Question : *calculer* $x^{(2)}$.
Solution :

$$y^{(0)} = (3,2), \;\; t_0 = 0.5 \;\; et \;\; x^{(1)}) = (0.5, 2.5),$$

$$y^{(1)} = (-5,2), \;\; t_1 = 0.13 \;\; et \;\; x^{(2)} = (-0.215, 2.435).$$

1.4 Méthode de gradient projeté

Pour rechercher la direction admissible en $x^{(k)}$ l'idée est de projeter le point $x^{(k+1)}$ sur l'ensemble S. Si S est particulier (un parallélépipède, une boule), on peut savoir calculer $S(x)$ et l'algorithme de gradient projeté est envisageable. Dans le cas général, on ne sait pas calculer $S(x)$ et il faut recourir à d'autres méthodes.

1.4.1 Construction de la méthode (Idée de Rosen)

Théorème 1.4.1. *(théorème de projection sur un convexe) Soit S un convexe fermé, et soit $v \in \mathbb{R}^n$, alors il existe un élément unique $P_S(v)$ tel que ,*

$$\| v - P_u(v) \| = \min_{u \in S} \| v - u \| . \tag{1.11}$$

On peut définir la suite des approximations $x^{(k)} \in S$ par,

$$x^{(k+1)}) = P_S(x^{(k)} + t_k d^{(k)}) = P_S(x^{(k)} - t_k \nabla^T f(x^{(k)})). \tag{1.12}$$

La projection essaye de garder $x^{(k+1)}$ dans S.

- **Déterminer** $d^{(k)}$
  Soit l'ensemble des indices des contraintes actives,

  $$I_k = \{i, a_i x^{(k)} = b_i\}.$$

  On sait qu'une direction d est admissible si,

  $$Ad = 0$$

  Alors on définit le sous-espace,

  $$P_0 = \{x \in \mathbb{R}^n, a_i^T x^{(k)}) = 0\}, \quad i \in I_k$$

  P_0 est le bord de la région admissible définie par ces contraintes.
  Dénotons A_k la matrice du rang $|I_k|$ dont les lignes sont les vecteurs a_i^T.
  Le sous espace P_v orthogonale à P_0 est,

  $$P_v = \{x \in \mathbb{R}^n, x = A_k^T v\}.$$

  Puisque $d^{(k)}$ est la projection de $-\nabla^T f(x^{(k)})$ sur P_0 alors,

  $$-\nabla^T f(x^{(k)}) = d^{(k)} + A_k^T v^k$$

  $$-A_k \nabla^T f(x^{(k)}) = A_k d^{(k)} + A_k A_k^T v^k$$

  Alors,

  $$v^k = -A_k (A_k A_k^T)^{-1} \nabla^T f(x^{(k)})$$

  Donc,

  $$d_k = -\nabla^T f(x^{(k)}) - A_k^T v^k = -[I - A_k (A_k A_k^T)^{-1} A_k] \nabla^T f(x^{(k)}).$$

- **Détermine t_k (pas optimal)**
  Une fois déterminer $d^{(k)}$ alors t_k est le pas optimal, c.-à-d.

$$t_k^* = \{t_k, f(x^{(k)} + t_k d^{(k)})\},$$

  Tel que,

$$t_k = \min_{i \notin I_k} \{\frac{b_i - a_i x^k}{a_i^T d^{(k)}}, a_i^T d^{(k)} > 0\}.$$

  Le critère d'arrêt de la méthode est que $d^{(k)} = 0$ et $v_k \geq 0$ (Le critère KKT).

1.4.2 Convergence de la méthode

On dit que f est de gradient supposé Lipchitzien de constante $L > 0$, si

$$\forall x, y \in \mathbb{R}^n, \| \nabla f(y) - \nabla f(x) \| \leq L \| y - x \| .$$

Proposition 1.4.1. *Si $0 < t_k < \frac{2}{L}$ alors, la suite $(x_k)_k$ générée par l'algorithme du gradient projeté converge vers un point qui minimise f.*

Note : Les méthodes en dessous ont été partagées comme des exposés au étudiants.

1.5 Algorithme d'Uzawa :

est une méthode itérative pour résoudre des problèmes d'optimisation avec contraintes non linéaires, en particulier des problèmes d'optimisation avec des contraintes d'inégalité. Il est souvent utilisé pour les problèmes d'optimisation convexes.

1.6 Méthode de Newton :

Utilise la matrice Hessienne pour trouver un minimum local en se basant sur une approximation quadratique de la fonction.

1.7 Méthode de Gauss-Newton :

Cette méthode est spécialement conçue pour résoudre des problèmes d'optimisation non linéaire liés à des modèles de moindres carrés non linéaires.

1.8 Méthode de Quasi-Newton :

Variante de la méthode de Newton qui approxime la Hessienne sans la calculer exactement. Exemple : BFGS.

1.9 Méthode de Gradient Conjugué :

Utilise le gradient pour trouver un minimum en exploitant la conjugaison de directions de recherche (fait en cours).

1.10 Algorithme de Levenberg-Marquardt :

Utilisé pour résoudre des problèmes d'ajustement de courbe, il combine des éléments de la méthode de Gauss-Newton et de la méthode de gradient stochastique.

1.11 Méthode de Marquardt non linéaire :

Une variante de l'algorithme de Levenberg-Marquardt utilisée pour minimiser la somme des carrés des résidus.

1.12 Méthode de Descente de Gradient :

Utilise le gradient de la fonction objectif pour trouver un minimum local en suivant la direction la plus raide.

1.13 Méthode de Descente de Gradient Stochastique (SGD) :

Utilisée pour l'optimisation de fonctions coût en apprentissage automatique et en apprentissage profond. Elle suit un échantillon aléatoire des données.

1.14 Méthode de Recuit Simulé (Simulated Annealing) :

Utilisée pour l'optimisation globale, elle simule le refroidissement d'un système physique pour éviter les minima locaux.

1.15 Méthode de l'Évolution Différentielle :

Utilise une population de solutions pour explorer l'espace de recherche et trouver un minimum global.

1.16 Méthode de l'Essaim de Particules (Particle Swarm Optimization) :

Inspirée du comportement social des oiseaux ou des poissons, elle guide un ensemble de "particules" vers la recherche d'un optimum.

1.17 Algorithme génétique :

Utilisé pour résoudre des problèmes d'optimisation en s'inspirant de la sélection naturelle et de l'évolution.

1.18 Méthode de la Fonction de Valeur Quadratique (Quadratic Approximation) :

Utilise une approximation quadratique de la fonction objectif autour d'un point.

1.19 Méthode de la Recherche Linéaire :

Trouve le minimum d'une fonction en effectuant une recherche le long d'une ligne dans l'espace des variables.

1.20 Méthode de la Région de Confiance (Trust Region) :

Utilise une région de confiance pour gérer la taille des pas lors de l'optimisation.

1.21 Méthode de la Pénalité Extérieure (Exterior Penalty Function) :

Utilisée pour résoudre des problèmes d'optimisation avec contraintes en convertissant les contraintes en termes de pénalités dans la fonction objectif.

1.22 Méthode de la Barrière (Barrier Method) :

Utilisée pour résoudre des problèmes d'optimisation avec contraintes en ajoutant une fonction de barrière dans la fonction objectif.

1.23 Méthode de l'Intérieur (Interior Point Method) :

Une approche pour les problèmes d'optimisation linéaire et non linéaire avec contraintes en utilisant des points intérieurs.

1.24 Méthode de Points Intérieurs Primal-Dual :

Cette méthode combine les méthodes duales et primales en utilisant une approche de point intérieur. Elle suit une trajectoire à l'intérieur de la région admissible tout en optimisant les contraintes duales.

1.25 Méthode de Pénalisation :

Cette méthode convertit les contraintes non linéaires en termes de pénalités dans la fonction objectif, créant ainsi un problème d'optimisation avec des contraintes pénalisées.

1.26 Méthode de Gradient Augmenté :

Cette méthode combine des itérations de gradient avec la résolution du problème dual pour trouver la solution optimale.