Chapitre 4

Méthode de gradient (pour des problèmes sans contrainte)

4.1 Introduction

Dans ce chapitre on s'intéresse à la résolution d'un problème d'optimisation convexe sans contrainte de la forme

$$f(x^\star) = \inf_{x \in \mathbb{R}^n} f(x)$$

où $f : \mathbb{R}^n \to \mathbb{R}$ est une fonction convexe et deux fois différentiable (sur $\mathbb{R}^n$). Sous ces hypothèses on sait que résoudre le problème d'optimisation revient à résoudre les n équations non linéaires

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x_1, \ldots, x_n) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x_1, \ldots, x_n) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

Afin d'assurer l'existence et l'unicité d'une telle solution, nous allons supposer que f est fortement convexe.

4.2 Méthode de gradient

4.2.1 Algorithme général de descente

Rappel 4.1 – Direction de descente/*Descent direction*/下降方向

Soit x tel que $\nabla f(x) \neq 0$. Le vecteur $d \in \mathbb{R}^n$ est une direction de descente au point x si et seulement si $\langle \nabla f(x), d \rangle \leqslant 0$.

Remarque 4.1 – Condition de résiliation

La condition de résiliation n'est pas unique. Pouvez-vous penser à d'autres conditions possibles ?

- Nombre d'itérations
- Critère de tolérance
- Objectif de précision
- Convergence de la fonction de coût

Convergence : Si on fait abstraction du critère d'arrêt, un algorithme de descente produit une suite de points $(x_k)_{k \in \mathbb{N}}$ définie par la relation de récurrence

$$x_{k+1} = x_k + \delta_k . d_k$$

Algorithm 1 Algorithme général de descente

Données : Un point initial $x_0 \in \mathbb{R}^n$, un seuil de tolérance $\varepsilon > 0$
Résultat : Un point $x \in \mathbb{R}^n$ proche de $x^\star$

Initialiser : $x \leftarrow x_0$
$k \leftarrow 0$
Fonction DESCENTE GÉNÉRAL
  Tant que $\|\nabla f(x)\| \geqslant \varepsilon$ **faire**
    1. Déterminer une direction de descente $d_k \in \mathbb{R}^n$.
    2. Déterminer un pas de descente $\delta_k > 0$ tel que $f(x_k + \delta_k.d_k) < f(x_k)$.
    3. Mettre à jour x : $x \leftarrow x_{k+1} = x_k + \delta_k.d_k$
    $k \leftarrow k + 1$
  Fin Tant que
  Retourner x
Fin Fonction

et telle que $f(x_{k+1}) < f(x_k)$ (sauf si $x_k = x^\star$ à partir d'un certain rang). L'étude de la convergence d'un tel algorithme de descente consiste donc à savoir si la suite $(x_k)_{k\in\mathbb{N}}$ converge vers $x^\star$. On rappelle que si f est fortement convexe on a $\|x - x^\star\| \leqslant \frac{2}{m}\|\nabla f(x)\|$, donc le critère d'arrêt $\|\nabla f(x)\| < \varepsilon$ implique $\|x - x^\star\| \leqslant \frac{2\varepsilon}{m}$.

Vitesse de convergence : Une fois qu'on a prouvé qu'un algorithme converge (par exemple $(x_k)_{k\in\mathbb{N}}$ converge vers $x^\star$), on s'intéresse à sa vitesse de convergence. On dit que la méthode est d'ordre $r \geqslant 1$, s'il existe une constante $C > 0$ telle que, pour k suffisamment grand

$$\frac{\|x_{k+1} - x^\star\|}{\|x_k - x^\star\|^r} \leqslant C$$

— Si $r = 1$, il faut $C \in]0, 1[$ pour avoir convergence et on a alors convergence linéaire.
— Si $r = 2$, on a une convergence quadratique.
— Si $\lim \frac{\|x_{k+1}-x^\star\|}{\|x_k-x^\star\|^r} = 0$ alors on dit que l'on a convergence superlinéaire (ce qui est le cas pour toutes les méthodes d'ordre $r > 1$).

Remarque 4.2

L'algorithme du descente à pas fixe est une méthode de descente utilisant un pas fixe

$$\forall k \in \mathbb{N},\ \delta_k = \delta$$

4.2.2 Algorithme de recherche de pas de descente

Dans l'Algorithme 1, l'étape «Déterminer un pas de descente $\delta_k > 0$ tel que $f(x_k + \delta_k.d_k) < f(x_k)$» nous demande de chercher un pas approprié. Alors, nous allons nous limiter à deux méthodes, la méthode de pas de descente optimale (dite aussi exacte) et la méthode de pas de descente par rebroussement.

Dans les deux cas, les données du problème sont une fonction f fortement convexe, un point actuel $x = x_k \neq x^\star$, une direction de descente $d = d_k$ pour le point x, et on cherche un pas de descente $\delta = \delta_k > 0$ tel que $f(x + \delta.d)$ soit «suffisamment plus petit» que $f(x)$.

Définition 4.1 – Méthode de pas de descente optimale

Pour f une fonction fortement convexe, $x \in \mathbb{R}^n$ et $d \in \mathbb{R}^n$ une direction de descente pour f en x, pour minimiser $f(x + \delta.d)$, la méthode optimale donne le pas de descente comme [a]

$$\delta^\star = \arg\min_{\delta>0} f(x + \delta.d)$$

a. $\arg\min f$ est l'ensemble des valeurs x pour lesquelles f atteint son minimum

$$\arg\min f \overset{\text{déf}}{=} \left\{x \in \mathbb{R}^n | \forall x' \in \mathbb{R}^n,\ f(x') \leqslant f(x)\right\}$$

Remarque 4.3

- Le pas de descente optimal est bien défini, c'est-à-dire que $\delta \mapsto f(x+\delta.d)$ admet un unique minimum global $\delta^\star$ sur $]0,+\infty[$.
- On a $\delta = \delta^\star$ si et seulement si $\langle \nabla f(x+\delta.d), d\rangle = 0$.

La méthode de rebroussement permet de calculer un pas de descente lorsque l'on ne sait pas minimiser la fonction f sur la demi-droite affine $\{x+\delta.d, t>0\}$.

Définition 4.2 – Méthode de pas de descente par rebroussement

Pour f une fonction fortement convexe, $x \in \mathbb{R}^n$ et $d \in \mathbb{R}^n$ une direction de descente pour f en x, il existe, un entier $N \in \mathbb{N}$, deux réels $\alpha \in]0,1[$ et $\beta \in]0,1[$ tels que si $n \geqslant N$, on a

$$f(x+\beta^n.d) \leqslant f(x) + \alpha\beta^n \langle \nabla f(x), d\rangle$$

La méthode de rebroussement donne le pas de descente comme

$$\delta^\star = \beta^N$$

4.2.3 Algorithme de descente de gradient

On s'intéresse maintenant à l'étude d'un algorithme de descente particulier appelé algorithme de descente de gradient. Cet algorithme utilise comme direction de descente $d = d_k$ au point $x = x_k$ le vecteur opposé du gradient, soit

$$d_k = -\nabla f(x_k)$$

Algorithm 2 Algorithme de descente

Données : Un point initial $x_0 \in \mathbb{R}^n$, un seuil de tolérance $\varepsilon > 0$
Résultat : Un point $x \in \mathbb{R}^n$ proche de $x^\star$

Initialiser : $x \leftarrow x_0$
$k \leftarrow 0$
Fonction DESCENTE DE GRADIENT
  Tant que $\|\nabla f(x)\| \geqslant \varepsilon$ **faire**
    1. Calculer $d_0 = -\nabla f(x)$ $(d_k = -\nabla f(x_k))$.
    2. Déterminer un pas de descente $\delta_k > 0$ par la méthode optimale ou par la méthode de rebroussement.
    3. Mettre à jour x : $x \leftarrow x_{k+1} = x_k + \delta_k.d_k$
    $k \leftarrow k+1$
  Fin Tant que
  Retourner x
Fin Fonction

Attention, en pratique, on teste plutôt le critère d'arrêt après l'étape 1 afin de ne pas calculer deux fois $\nabla f(x_k)$. La convergence de l'algorithme est assurée par le théorème suivant.

Théorème 4.1 – Convergence de la méthode de gradient

Soient $f : \mathbb{R}^n \to \mathbb{R}$ une fonction fortement convexe telle que

$$\forall (x,h) \in (\mathbb{R}^n)^2,\ m\|h\|^2 \leqslant \langle H_f(x)\cdot h, h\rangle \leqslant M\|h\|^2$$

avec $0 < m \leqslant M$ et x_0 un point quelconque de $\mathbb{R}^n$. Alors l'algorithme de descente de gradient converge et on a une vitesse de convergence linéaire

$$\forall k \in \mathbb{N},\ f(x_k) - f(x^\star) \leqslant c^k (f(x_0) - f(x^\star))$$

où $c \in [0,1[$ dépend de la méthode de recherche de pas de descente et est donnée par

— $c = 1 - \frac{m}{N}$ *pour la méthode exacte/optimale,*

— $c = 1 - \min\left(2m\alpha, 2\beta\alpha\frac{m}{M}\right)$ *pour la méthode de rebroussement utilisant les constantes* $\alpha \in]0, \frac{1}{2}[$ *et* $\beta \in]0, 1[$.

Démonstration 10

On peut démontrer qu'avec la condition $m\|h\|^2 \leqslant \langle H_f(x)\cdot h, h\rangle \leqslant M\|h\|^2$ $(0 < m \leqslant M)$,

$$\frac{2}{m}(f(x_k) - f(x^\star)) \leqslant \|x_k - x^\star\|^2 \leqslant \frac{2}{M}(f(x_k) - f(x^\star))$$

ce qui montre que la convergence de $(f(x_k))_{k\in\mathbb{N}}$ vers $f(x^\star)$ implique la convergence de $(x_k)_{k\in\mathbb{N}}$ vers $x^\star$.

— **La preuve de la convergence linéaire dans le cas de la méthode optimale pour le calcul du pas de descente.**
On suppose que l'algorithme de gradient est à l'itération k et que $x_k \neq x^\star$ (sinon l'algorithme a convergé en un nombre fini $l \leqslant k$ d'itérations et la suite (x_k) est constante et égale à $x^\star$ à partir du rang l). Le point x_{k+1} est de la forme $x_{k+1} = x_k + \delta_k.\nabla f(x_k)$ avec

$$\delta_k = \arg\min_{\delta>0} f(x_k + \delta.\nabla f(x_k))$$

Pour tous $(x, y) \in (\mathbb{R}^n)^2$,

$$f(y) \leqslant f(x) + \langle \nabla f(x), y - x\rangle + \frac{M}{2}\|y - x\|^2$$

Alors, en prenant $x = x_k$ et $y = x_k - \delta.\nabla f(x_k)$ pour tout $\delta > 0$ on a

$$\begin{aligned} f(x_k - \delta.\nabla f(x_k)) &\leqslant f(x_k) + \langle \nabla f(x_k), -\delta.\nabla f(x_k)\rangle + \frac{M}{2}\|-\delta.\nabla f(x_k)\|^2 \\ &= f(x_k) - \delta\|\nabla f(x_k)\|^2 + \delta^2\frac{M}{2}\|\nabla f(x_k)\|^2 \end{aligned}$$

Par définition, le membre de gauche est minimal en $\delta = \delta_k$. Ainsi pour tout $\delta > 0$, on a

$$f(x_{k+1}) \leqslant f(x_k) + \left(\delta^2\frac{M}{2} - \delta\right)\|\nabla f(x_k)\|^2$$

Le membre de droite est minimal en $\delta = \frac{1}{M}$, et pour cette valeur de δ on obtient la majoration

$$f(x_{k+1}) \leqslant f(x_k) - \frac{1}{2M}\|\nabla f(x_k)\|^2$$

On soustrait ensuite la valeur minimum $f(x^\star)$ à cette inégalité

$$f(x_{k+1}) - f(x^\star) \leqslant f(x_k) - f(x^\star) - \frac{1}{2M}\|\nabla f(x_k)\|^2$$

Enfin, par forte convexité, pour tout $x \in \mathbb{R}^n$,

$$f(x^\star) \geqslant f(x) - \frac{1}{2m}\|\nabla f(x)\|^2$$

et donc pour $x = x_k$,

$$\|\nabla f(x_k)\|^2 \geqslant 2m(f(x_k) - f(x^\star))$$

Ainsi,

$$f(x_{k+1}) - f(x^\star) \leqslant \left(1 - \frac{m}{M}\right)(f(x_k) - f(x^\star))$$

Par récurrence, on a donc bien $f(x_k) - f(x^\star) \leqslant c^k(f(x_0) - f(x^\star))$ avec $c = 1 - \frac{m}{M} \in [0, 1[$.

— **La preuve de la convergence linéaire dans le cas de la méthode de rebroussement.**

(1) Montrer que pour tout $\delta \in [0, \frac{1}{M}]$ on a $\frac{M}{2}\delta^2 - \delta \leqslant -\frac{t}{2}$ (on pourra par exemple utiliser la convexité).
(2) En déduire que pour tout $\delta \in [0, \frac{1}{M}]$,

$$f(x_k + \delta.\nabla f(x_k)) \leqslant f(x_k) - \alpha\delta\|\nabla f(x_k)\|^2$$

(3) En déduire que la méthode de rebroussement s'arrête soit en $\delta_k = 1$ soit pour une valeur $\delta_k \geqslant \frac{\beta}{M}$.
(4) En déduire que

$$f(x_{k+1}) \leqslant f(x_k) - \min\left(\alpha, \frac{\alpha\beta}{M}\right)\|\nabla f(x_k)\|^2$$

et conclure la preuve comme pour le cas du pas optimal.

En pratique, l'algorithme de gradient à pas optimal s'avère souvent plus efficace que l'algorithme de gradient à pas fixe

Exemple 4.1 – L'algorithme de gradient à pas optimal est plus efficace

La matrice hessienne de la forme quadratique $f(x,y) = x^2 + 100y^2$ est la matrice diagonale constante $H_f = \begin{bmatrix} 2 & 0 \\ 0 & 200 \end{bmatrix}$, alors $0 < 2.Id \leqslant H_f \leqslant 200.Id$. Donner le nombre d'itérations nécessaires pour approcher le minimum $(0,0)$ de f sur $\mathbb{R}^2$ à 10^{-6} près, à partir de l'initialisation : $(x_0, y_0) = (1,1)$.

- **à pas fixe**
  Lorsque la matrice hessienne est mal conditionnée comme $\frac{M}{m} = \frac{200}{2} \gg 1$, l'algorithme est lent. Le taux de convergence optimal est obtenu pour un pas égal à : $\frac{1}{101} \approx 0.0099$. Pour cette valeur du pas, il faut près de 700 itérations pour approcher le minimum $(0,0)$ de f à 10^{-6} près. Si le pas est supérieur à 0.01, l'algorithme diverge.
- **à pas optimal**
  Le paramètre de tolérance ε utilisé pour la phase de recherche linéaire, étant fixé à 10^{-8}, il ne faut que 6 itérations pour approcher le minimum $(0,0)$ de f à 10^{-6} près.

Il est cependant difficile de comparer objectivement les deux algorithmes, et le contre-exemple suivant montre l'impossibilité d'établir théoriquement, et pour toute initialisation donnée, la supériorité de l'algorithme du gradient à pas optimal, même lorque le critère est une forme quadratique elliptique simple :

Exemple 4.2 – Mais pas toujours plus efficace ...

Si l'on cherche à minimiser $f = x^2 + 2y^2$ sur $\mathbb{R}^2$ à 10^{-6} près, a partir de toute initialisation (x_0, y_0) située sur la droite d'équation : $x = 2y$. Le pas optimal effectué à chaque étape est constant égal à $1/3$, et la vitesse de convergence linéaire de taux : $1/3$.

4.3 Application de la méthode de gradient

Exemple 4.3 – Méthode de gradient

Pour $(x,y) \in \mathbb{R}^n$, minimiser la fonction

$$f(x,y) = \frac{1}{2}\left((x+y-4)^2 + (2x+3y-7)^2 + (4x+y-9)^2\right)$$

On commence par définir la fonction f avec les variables x et y

In[1]

```
def fonctionf(x, y):
    f = ((x+y-4)**2 + (2*x+3*y-7)**2 + (4*x+y-9)**2)*0.5
    return f
```

Ensuite, les dérivées partielles du premier ordre de la fonction f en termes de x et y

In[2]

```
import sympy as sp
def deriv(x, y):
    a , b = sp.symbols('a b', real = True)
    fx = sp.diff(fonctionf(a, b), a).subs(a,x).subs(b,y)
    fy = sp.diff(fonctionf(a, b), b).subs(a,x).subs(b,y)
```

```
    return fx, fy
```

Les dérivés partielles $\frac{\partial f}{\partial x}$ et $\frac{\partial f}{\partial y}$ sont définies respectivement par `fx` et `fy`.

$$\begin{cases} \dfrac{\partial f}{\partial x}(x,y) = (x+y-4) + (2x+3y-7) \times 2 + (4x+y-9) \times 4 \\ \dfrac{\partial f}{\partial y}(x,y) = (x+y-4) + (2x+3y-7) \times 3 + (4x+y-9) \end{cases}$$

La méthode de gradient à pas fixe.
Premièrement, on présent une méthode de calcul itératif de gradients avec un pas fixe $\delta = 0.01$, à partir d'un point initial $(x, y) = (0, 0)$. La condition pour abandonner l'opération est que la valeur de la fonction ne diminue plus, ou que la différence entre les deux valeurs de fonction calculées soit inférieure à 10^{-6}.

In[3]

```
# n = nombre maximal d'itérations
def gradient_pas_fixe(n):
    delta = 0.01    # pas fixe
    x, y = 0, 0     # point initial (0, 0)
    f = fonctionf(x, y)
    deriv_x, deriv_y = deriv(x, y)
    m = 0
    # Condition de résiliation
    while m < n and deriv_x**2+deriv_y**2 > 1e-6 :
        m += 1
        deriv_x, deriv_y = deriv(x, y)
        x = x - delta * deriv_x
        y = y - delta * deriv_y
        f = fonctionf(x, y)
    return x, y, f, m
```

Le résultat d'une itération de 10, 100 et 1000 fois

In[4]

```
gradient_pas_fixe(10)
# (1.86918954756589, 1.13873258547825, 0.577979920341783, 10)

gradient_pas_fixe(100)
# (1.99835128077227, 1.09346955876043, 0.454563617991453, 100)

gradient_pas_fixe(1000)
# (1.99987219745699, 1.09110756879253, 0.454545563685425, 164)
```

Exemple 4.4

Nous prenons un exemple intuitif pour voir l'effet des tailles de pas fixe sur les résultats d'optimisation.

En observant les trois images, on peut constater que :

1. Dans l'image de gauche, le taux d'apprentissage est trop faible. En conséquence, la solution ne peut pas être trouvée dans les dix premières étapes. Cependant, si l'algorithme continue à itérer sur une longue période, il peut éventuellement trouver la solution.
2. Dans l'image du milieu, le taux d'apprentissage semble parfaitement adapté. En quelques itérations, l'algorithme converge rapidement vers la solution finale.

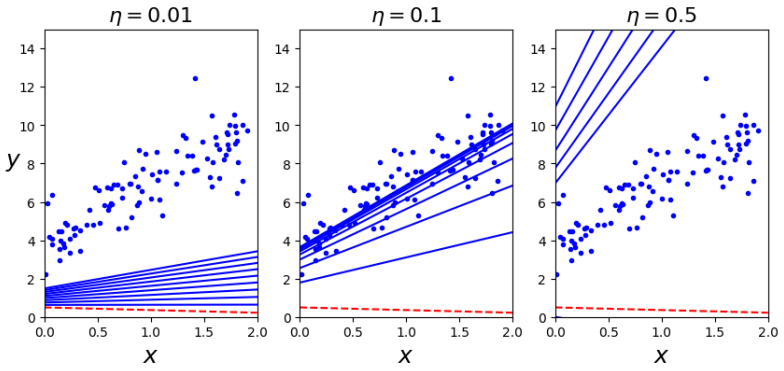

FIGURE 4.1 – Les premières 10 itérations d'optimisation avec les pas fixes $\eta = 0,01$, $\eta = 0,1$ et $\eta = 0,5$.

3. À droite, le taux d'apprentissage est trop élevé. L'algorithme diverge, saute la région de données et s'éloigne progressivement de la solution réelle à chaque étape.

Ces exemples montrent l'importance de choisir une taille de pas adaptée pour l'apprentissage automatique. Lorsque cette taille de pas ne peut pas être prédéterminée, il est nécessaire d'utiliser d'autres méthodes de sélection.

La méthode de gradient à pas optimal.

L'idée de trouver un pas approprié δ_k à chaque étape est de résoudre l'équation

$$\langle \nabla f(x + \delta_k.d), d \rangle = 0$$

Ici, on utilise la fonction `sympy.solve()` en Python pour obtenir la solution de l'équation. Cette fonction ne peut parfois pas obtenir de solution analytique, et on peut alors utiliser la fonction de calcul numérique `sympy.nsolve()` pour approximer la solution. Cependant, il peut y avoir des erreurs dans la solution numérique, ce qui est un problème majeur avec Python.

In[3]

```
import numpy
from sympy import *
def pas_optimal(x, y):
    delta = symbols('delta')
    deriv_x, deriv_y = deriv(x, y)
    x_n = x - delta * deriv_x
    y_n = y - delta * deriv_y
    fonc = numpy.dot(deriv(x_n, y_n), deriv(x, y))
    pas = solve(fonc, delta)
    return pas[0]
```

Le pas ici changera en fonction de l'itération de calcul.

In[4]

```
def gradient_pas_optimal(n):
    x, y = 0, 0      # point initial (0, 0)
    f = fonctionf(x, y)
    deriv_x, deriv_y = deriv(x, y)
    m = 0
    # Condition de résiliation
    while m < n and deriv_x**2+deriv_y**2 > 1e-6 :
        m += 1
        delta = pas_optimal(x, y)     # pas optimal
        deriv_x, deriv_y = deriv(x, y)
```

```
        x = x - delta * deriv_x
        y = y - delta * deriv_y
        f = fonctionf(x, y)
    return x, y, f, m
```

Le résultat d'une itération de 1, 10 et 100 fois

In[5]

```
gradient_pas_optimal(1)
# (1.92303924998251, 1.21080249072973, 0.494297908066886, 1)

gradient_pas_optimal(10)
# (1.99999939946619, 1.09090876334520, 0.454545454551995, 4)
```

On peut voir que cette méthode de descente de gradient à pas variable a obtenu un résultat stable après 10 itérations. Même s'il existe de légères différences par rapport aux résultats de la méthode à pas fixe, on peut encore conclure que cette méthode est plus efficace pour cet exemple.

4.4 Exercices

4.1 Supposons que vous avez une fonction de coût $f(x, y) = x^2 + 2y^2 - 2x - 4y + 1$ définie sur $\mathbb{R}^2$. Utilisez la méthode de descente de gradient pour trouver le minimum de cette fonction.

4.2 Considérez la fonction de coût suivante $J(w) = w^2 - 6w + 5$, avec $w \in \mathbb{R}$. Appliquez la méthode de descente de gradient pour trouver le minimum de la fonction en partant de $w = 0$.

a. Utilisez un taux d'apprentissage (pas fixe) de $0,1$.

b. Résolvez ce problème avec la méthode de descente à pas optimal.

c. Comparez les deux résultats.

4.3 (Méthode de Newton, voir l'Annexe A) Trouver la direction de descente pour la fonction de coût $f(x) = x^2+y^2$ en $(1, 1)$ en utilisant l'algorithme de descente de gradient avec une précision de $0,001$.

4.4 (Méthode de Newton, voir l'Annexe A) Utilisez la méthode de Newton pour trouver le minimum de la fonction $f(x) = x^3 - 3x^2 + 2$ en partant de $x_0 = 0,5$ avec une précision de $0,001$.