



Étudiants ingénieurs en aérospatial

Mémoire de 3^e année

Théorie et Benchmark des méthodes de descente en vue d'une application au machine learning

Auteurs :

M. AUDET Yoann

M. CHANDON Clément

M. DE CLAVERIE Chris

M. HUYNH Julien

Encadrant :

Pr. PESCHARD Cédric

Version 1.0 du
5 mai 2019

Remerciements

Table des matières

1	Introduction	1
2	Développement théorique des méthodes de descente	2
2.1	Retour sur les méthodes de descente	2
2.1.1	Motivation de l'étude et de l'intérêt	2
2.1.2	Les méthodes de gradients	2
2.2	Introduction aux espaces de Krylov	2
2.2.1	Définition des espaces de Krylov	2
2.2.2	Quelques propriétés	3
2.2.3	Un premier algorithme : GMRES	4
2.3	L'approche par les espaces de Krylov du gradient conjugué	8
2.3.1	Définition du gradient conjugué	8
2.3.2	Propriétés de l'implémentation du gradient conjugué	9
2.3.3	Algorithme du gradient conjugué	12
2.3.4	Quelques mots sur le gradient conjugué non-linéaire	12
2.4	D'autres algorithmes de minimisation	12
2.4.1	Les algorithmes de Newton	12
2.4.2	Les algorithmes de Quasi-Newton	13
3	Comparaison logiciel des méthodes	14
4	Applications des méthodes de descente au Machine Learning	15
4.1	Introduction et Motivation	15
4.2	Application des algorithmes	15
5	Conclusion	16

Chapitre 1

Introduction

Chapitre 2

Développement théorique des méthodes de descente

Le but de ce rapport est d'étudier les différentes possibilités afin de minimiser une fonction. Le problème se formule de la manière suivante :

$$\operatorname{argmin}_x f(x) \tag{2.1}$$

On cherche donc l'argument x tel que $f(x)$ soit minimal.

2.1 Retour sur les méthodes de descente

2.1.1 Motivation de l'étude et de l'intérêt

2.1.2 Les méthodes de gradients

Les méthodes de gradients sont des méthodes de descentes.

2.2 Introduction aux espaces de Krylov

Dans cette partie, nous allons introduire un outil mathématique important qui permet la justification de la méthode du gradient conjugué : les espaces de Krylov.

2.2.1 Définition des espaces de Krylov

Afin d'introduire les espaces de Krylov, nous allons prendre un exemple simple : celui d'un système linéaire. On cherche donc x qui vérifie l'équation :

$$Ax = b \tag{2.2}$$

On définit le vecteur résidu à l'itération k comme étant :

$$r^k \triangleq b - Ax^k = -A(-A^{-1}b + x^k) = -A(-x^* + x^k) \quad (2.3)$$

Où x^* est la solution réelle du système. Alors, nous pouvons écrire la solution au rang $k+1$, comme celle au rang k plus le résidu :

$$x^{k+1} = x^k + r^k \quad (2.4)$$

$$\Leftrightarrow x^{k+1} - x^* = x^k - x^* + r^k \quad (2.5)$$

$$\Leftrightarrow -A(x^{k+1} - x^*) = -A(x^k - x^*) - Ar^k \quad (2.6)$$

$$\Leftrightarrow r^{k+1} = r^k - Ar^k \quad (2.7)$$

Dans cette dernière équation récursive, nous pouvons voir que r^{k+1} est une combinaison linéaire des vecteurs r^k précédents. Ainsi :

$$r^k \in Vect\{r^0, Ar^0, \dots, A^k r^0\} \quad (2.8)$$

Cela implique directement :

$$x^k - x^0 = \sum_{i=0}^{k-1} r^i \quad (2.9)$$

Donc il vient que :

$$x^k \in x^0 + Vect\{r^0, Ar^0, \dots, A^k r^0\} \quad (2.10)$$

Où $Vect\{r^0, Ar^0, \dots, A^k r^0\}$ est le k -ième espace de Krylov généré par A à partir de r^0 noté $\mathcal{K}_k(A, r^0)$

2.2.2 Quelques propriétés

Maintenant que nous avons une définition de ces espaces, nous allons montrer plusieurs propriétés pour ensuite construire l'algorithme et l'implémenter. Le premier fait remarquable des espaces de Krylov est que par construction, nous avons :

$$\mathcal{K}_{k-1}(A, r^0) \in \mathcal{K}_k(A, r^0) \quad (2.11)$$

Ensuite, il est important de montrer que la solution que l'on cherche appartient à cet espace. Nous reprenons donc notre système linéaire possédant une unique solution :

$$Ax = b \quad (2.12)$$

D'après la définition du problème, la matrice A est inversible. Nous supposons que l'on a le polynôme caractéristique de A :

$$P(\lambda) = \sum_{j=0}^n \alpha_j \lambda^j \Rightarrow P(0) = \alpha_0 = \det(A) \neq 0 \quad (2.13)$$

Par le théorème de Cayley-Hamilton¹, nous pouvons obtenir la valeur de A^{-1} :

$$P(A) = \alpha_0 I + \alpha_1 A + \dots + \alpha_n A^n = 0 \quad (2.14)$$

$$\alpha_0 A^{-1} A + \alpha_1 A + \dots + \alpha_n A^n = 0 \quad (2.15)$$

$$(\alpha_0 A^{-1} + \alpha_1 + \dots + \alpha_n A^{n-1}) A = 0 \quad (2.16)$$

$$\alpha_0 A^{-1} + \alpha_1 + \dots + \alpha_n A^{n-1} = 0 \quad (2.17)$$

Ce qui donne finalement :

$$A^{-1} = -\frac{1}{\alpha_0} \times \sum_{j=0}^{n-1} \alpha_{j+1} A^j \quad (2.18)$$

Or, la solution du problème 2.12 est : $x^* = A^{-1}b$. Ce qui peut s'écrire de la façon suivante :

$$x^* = -\frac{1}{\alpha_0} \sum_{j=0}^{n-1} \alpha_{j+1} A^j b \quad (2.19)$$

Ce vecteur appartient clairement à l'espace de Krylov défini par :

$$x^* \in \mathcal{K}(A, b) \quad (2.20)$$

2.2.3 Un premier algorithme : GMRES

Un des algorithmes utilisant les espaces de Krylov est l'algorithme GMRES qui se trouve ci-dessous en python. Cette méthode repose sur les espaces de Krylov. En effet, dans cet algorithme nous choisissons $x_k \in \mathcal{K}_k(A, b)$ tel que l'on a $\|b - Ax_k\|_2$ qui est minimal. Nous nous reposons sur une méthode de projection. L'algorithme utilise aussi sur le procédé d'Arnoldi² qui nous permet de créer une base orthonormée sur le sous-espace de Krylov. Dans la suite de ce chapitre, nous allons étudier ces composantes permettant d'arriver à l'écriture de notre algorithme.

1. http://www.logique.jussieu.fr/~alp/Cayley_Hamilton.pdf

2. <https://docplayer.fr/215470-Calcul-matriciel-et-systemes-lineaires.html> (page 36)

Généralité

Nous avons le système linéaire suivant qui possède une unique solution x^* :

$$Ax = b \quad (2.21)$$

Le résidu initial est :

$$r_0 = b - Ax^{(0)} \quad (2.22)$$

L'algorithme GMRES repose sur une méthode de projection. On écrit donc le système sous sa forme résiduelle pour faire apparaître une projection :

$$\langle b - Ax^*, v \rangle = 0, \forall v \in \mathbb{R}^n \Leftrightarrow b - Ax^* \perp \mathbb{R}^n \quad (2.23)$$

Le but est maintenant de construire des solutions par itérations. On se place dans un sous-espace \mathbf{K} et on essaie de trouver dans ce sous-espace x tel que :

$$b - Ax \perp \mathbf{K} \Rightarrow \langle Au, v \rangle = \langle b, v \rangle, v \in \mathbf{K} \quad (2.24)$$

On généralise cette méthode grâce à deux sous-espaces : \mathbf{K} et \mathbb{L} . On cherche $x \in u_0 + \mathbf{K}$ avec :

$$b - Ax \perp \mathbb{L} \Rightarrow \langle Au, v \rangle = \langle b, v \rangle, v \in \mathbb{L} \quad (2.25)$$

C'est la condition de Petrov-Galerkin.

Sous la forme matricielle, nous allons créer deux matrices V et W , de taille $n \times m$, définies par les vecteurs des bases de \mathbf{K} et \mathbb{L} :

$$V = (V_1 \ V_2 \ \dots \ V_m) \quad (2.26)$$

$$W = (W_1 \ W_2 \ \dots \ W_m) \quad (2.27)$$

Soit $(y, z) \in (\mathbb{R}^m)^2$. Nous pouvons définir $v \in \mathbf{K}$ et $w \in \mathbb{L}$:

$$v = Vy \quad (2.28)$$

$$w = Wz \quad (2.29)$$

Il nous faut maintenant trouver la solution de $x = x_0 + Vy \in \mathbf{K}$. Alors, grâce à 2.23 et 2.25, nous écrivons :

$$\langle Av, w \rangle = \langle r_0, w \rangle \ \forall w \in \mathbb{L} \quad (2.30)$$

$$\langle AVy, Wz \rangle = \langle r_0, Wz \rangle \ \forall z \in \mathbb{R}^m \quad (2.31)$$

$$Avy = r_0 \quad (2.32)$$

$$y = (W^t AV)^{-1} W^t r_0 \quad (2.33)$$

Nous obtenons donc la solution :

$$x = x_0 + V(W^t AV)^{-1} W^t r^0 \quad (2.34)$$

Nous pouvons donc maintenant expliquer ce que notre méthode de projection va faire. Il faut savoir qu'une méthode de projection est itérative et va à chaque itération faire deux choses :

- construire les sous-espaces K et L
- chercher la nouvelle itération dans $u + K$ par le procédé exposé ci-dessus.

Voici donc le principe général des méthodes de projection :

1. Choisir \mathbf{K}^m et \mathbb{L}^m : deux sous-espaces de \mathbb{R}^n de même dimension m
2. Construire V_m et W_m
3. Calculer le résidu : $r = b - Ax^{(k)}$
4. Résoudre le système $y = (W_m^t AV_m)^{-1} W_m^t r$
5. Créer la nouvelle solution : $x = x + V_m y$

Pour créer le sous-espace L , il y a deux méthodes :

- $L = K$ donne lieu à la méthode du gradient conjugué (expliqué ci-dessous)
- $L = AK$ donne lieu à la méthode GMRES.

La méthode du gradient conjugué ayant été vue en cours, nous avons choisi la deuxième méthode de génération de L . Dans les deux cas cela revient à résoudre le problème des moindres carrés :

$$\begin{cases} \text{On cherche } x^{(k)} \in x_0 + \mathbf{K}_m \text{ tel que :} \\ \min_{v \in \mathbf{K}_m} \frac{1}{2} \|A(x_0 + v) - b\|_2^2 \end{cases} \quad (2.35)$$

La démonstration liant les deux problèmes ne sera pas réalisée ici.³

Explication de l'algorithme GMRES

Cet algorithme se base dans un premier temps sur les itérations d'Arnoldi. Ici nous ne présenterons que la version qui est moins sensible aux erreurs d'arrondi car notre but est de justifier l'implémentation numérique de l'algorithme. Ce procédé d'Arnoldi a pour but de créer une base orthonormée du sous-espace de Krylov. Le procédé se résume par :

1. $v_1 = \frac{r_0}{\|r_0\|}$
2. Pour $j = 1, \dots, m$; on calcule : $w = Av_j$
3. Pour $i = 1, \dots, j$; on calcule : $w = w - (w, v_i)v_i$
4. $\bar{v}_{j+1} = \frac{w}{\|w\|}$

3. Elle se trouve ici : https://www2.mat.ulaval.ca/fileadmin/Cours/MAT-17992/notes_gmres.pdf

Nous introduisons alors un objet : $\overline{H}_m = (h_{ij})$ la matrice de Hessenberg :

$$h_{ij} = \begin{cases} \langle Av_j, v_i \rangle, & i \leq j \\ \|v_{j+1}\|, & i = j + 1 \\ 0, & \text{sinon} \end{cases} \quad (2.36)$$

Cet objet permet d'écrire de façon matricielle notre procédé car :

$$AV_m = V_{m+1}\overline{H}_m \quad (2.37)$$

et :

$$\bar{v}_{j+1} = Av_j - \sum_{i=1}^j h_{i,j}v_i \quad (2.38)$$

$$\Leftrightarrow Av_j = \bar{v}_{j+1} + \sum_{i=1}^j h_{i,j}v_i \quad (2.39)$$

$$\Leftrightarrow Av_j = h_{j+1,j}v_{j+1} + \sum_{i=1}^j h_{i,j}v_i \quad (2.40)$$

Par ce procédé, nous arrivons à construire une base orthonormée de l'espace de Krylov.

Maintenant que nous avons cette base, nous pouvons résoudre le problème des moindres carrés exprimé ci-dessus (cf. équation 2.35). Nous obtenons les propriétés suivantes (qui ne seront pas démontrées) :

- $\|r_{m+1}\| \leq \|r_m\|$
- L'algorithme converge en n itérations au maximum car $\mathcal{K}_m(A, r_0) = \mathbb{R}^n$

Voici le code et le pseudo-code de l'algorithme :

```

1 def GMRES(A, b, epsilon):
2     max_iter = A.shape[0] #Number of maxiter
3     mat_q = np.zeros((max_iter, max_iter + 1))
4     mat_h = np.zeros((max_iter + 1, max_iter))
5     norm_b = np.linalg.norm(b)
6     be1 = np.zeros(max_iter + 1)
7     be1[0] = norm_b
8     mat_q[:, 0] = 1 / np.linalg.norm(b) * b.T # On définit ici que l'on a
9     ↪ forcément x0 = 0 (r0 = b)
10    for j in range(max_iter):
```

4. La démonstration se trouve ici : <https://www.math.kth.se/na/SF2524/matber15/gmres.pdf>

```

10 mat_q[:, j+1] = A @ mat_q[:, j]
11 for i in range(j+1):
12     mat_h[i, j] = mat_q[:, i] @ mat_q[:, j + 1]
13     mat_q[:, j+1] -= mat_h[i, j] * mat_q[:, i]
14     mat_h[j + 1, j] = np.linalg.norm(mat_q[:, j + 1])
15     mat_q[:, j + 1] /= mat_h[j + 1, j]
16     y = np.linalg.lstsq(mat_h, be1, rcond=None)[0]
17     residue = np.linalg.norm(y) / norm_b
18     if residue < epsilon:
19         return mat_q[:max_iter, :max_iter] @ y, residue
20     return mat_q[:max_iter, :max_iter] @ y

```

2.3 L'approche par les espaces de Krylov du gradient conjugué

2.3.1 Définition du gradient conjugué

Grâce aux espaces de krylov, nous sommes capable d'améliorer les algorithmes de gradient qui sont présentés ci-dessus. Nous partons donc d'un algorithme du gradient basique :

$$\begin{cases} x_0 \in \mathbb{R} \text{ le choix initial} \\ x_{k+1} = x_k + \alpha_k(b - Ax_k) \end{cases} \quad (2.41)$$

Nous définissons alors le vecteur résidu $r_k = b - Ax_k$ appartenant à l'espace de Krylov d'ordre k et définit par le résidu à l'origine : r_0 . De ce fait, il vient que x_{k+1} appartient à l'espace affine composé par le point x_0 et le k^{ieme} espace de krylov : \mathcal{K}_k . Afin de simplifier notre exposé, nous supposons que nous avons affaire à des matrices symétriques définies positives.

Dans la méthode du gradient conjugué, nous n'allons pas choisir la définition comme dans les autres méthodes de gradient mais nous avons d'autres critères qui sont plus intéressants :

— Le premier repose sur le principe d'orthogonalisation :

$$\exists x_{k+1} \in [x_0 + \mathcal{K}_k], r_{k+1} \perp \mathcal{K}_k \quad (2.42)$$

— Le principe de minimisation :

$$\exists x_{k+1} \in [x_0 + \mathcal{K}_k], \frac{1}{2} \langle Ax, x \rangle - \langle b, x \rangle \text{ est minimal} \quad (2.43)$$

Dans le cas d'une matrice symétrique définie positive, les deux choix ci-dessus donnent la même solution x_{k+1} d'où le choix de A symétrique définie positive.

Définissons ensuite le vecteur direction :

$$d_k = x_{k+1} - x_k \quad (2.44)$$

De part la construction de d , il est possible de déduire quelques propriétés :

— L'espace de de Krylov est une combinaison linéaire des vecteurs r_k et d_k

$$\mathcal{K}_k = \text{vect} r_0, \dots, r_k = \text{vect} d_0, \dots, d_k \quad (2.45)$$

— Tous les vecteurs de la suite sont orthogonaux :

$$\forall 0 \leq l < k \leq n-1, \langle r_k, r_l \rangle = 0 \quad (2.46)$$

— Les vecteurs de la suite (d_k) sont conjugué selon le produit vectoriel défini par A :

$$\forall 0 \leq l < k \leq n-1, \langle A d_k, d_l \rangle = 0 \quad (2.47)$$

C'est grâce à cette dernière propriété que le gradient conjugué est appelé ainsi.

2.3.2 Propriétés de l'implémentation du gradient conjugué

Le gradient conjugué est aujourd'hui très utilisé dans différents domaines car c'est un algorithme possédant des propriétés très intéressantes. La première d'entre-elles concerne le calcul des vecteurs de direction. En effet, nous avons vu plus haut, lors de la construction de la méthode, que chaque vecteur est un conjugué de chacun des vecteurs précédents. C'est une propriété importante lors de l'implémentation logiciel de la technique puisque pour calculer le vecteur d_k nous n'avons besoin que de d_{k-1} .

Pour toutes méthodes numériques, il nous est important de montrer son utilité en étudiant la convergence : une méthode qui diverge est inutilisable et une méthode qui converge très lentement n'est guère plus utile. C'est pour cela que dans la suite de cette partie nous nous intéressons aux taux de convergence de notre gradient conjugué.

Tout d'abord, nous rappelons que le gradient conjugué converge en n itération au maximum. En effet, la solution de notre problème appartient à l'espace de Krylov dans lequel nous cherchons la solution. La convergence n'est donc pas un problème. Ce qui peut poser des problèmes c'est la vitesse à laquelle nous allons vers cette solution. Pour cela, nous partons de l'expression de x_{K+1} :

$$x_{k+1} = x_0 + \alpha_0 p_0 + \dots + \alpha_k p_k \quad (2.48)$$

$$= x_0 + \gamma_0 r_0 + \gamma_1 A r_0 + \dots + \gamma_k A^k r_0 \quad (2.49)$$

Pour trouver ces formules, nous appliquons juste la définition du gradient : nous partons d'un point x_0 et à chaque itération k nous avançons dans la direction p_k avec un pas de α_k .

Dans cette expression on reconnaît un polynôme dépendant de A :

$$P_k(A) = \gamma_0 I + \gamma_1 A + \dots + \gamma_k A^k \quad (2.50)$$

Ce qui simplifie l'expressions de x_{k+1} en :

$$x_{k+1} = x_0 + P_k(A)r_0 \quad (2.51)$$

En utilisant la norme définie par le produit scalaire avec la matrice A , nous avons :

$$\frac{1}{2} \|x - x^*\|_A^2 = \frac{1}{2} (x - x^*)^T A (x - x^*) \quad (2.52)$$

$$= \frac{1}{2} (x^T A x + x^{*T} A x^* - x^T A x^* - x^{*T} A x) \quad (2.53)$$

Or par définition, A est symétrique, $b = Ax^*$ et en utilisant la propriété de commutation du produit scalaire, nous avons :

$$\frac{1}{2} \|x - x^*\|_A^2 = f(x) - f(x^*) \quad (2.54)$$

avec $f(x) = \frac{1}{2} x^T A x - b^T x$. À chaque nouvelle itération nous allons donc minimiser cette expression pour tout x habitant dans l'espace de Krylov :

$$\min_{P_k} \|x_0 + P_k(A)r_0 - x^*\|_A \quad (2.55)$$

En introduisant cette minimisation dans l'expression du résidu et le fait que $r_0 = A(x_0 - x^*)$, il vient :

$$x_{k+1} - x^* = x_0 + P_k^*(A)r_0 - x^* = [I + P_k^*(A)A](x_0 - x^*) \quad (2.56)$$

On décompose A grâce à ses valeurs propres :

$$A = \sum_{i=1}^n \lambda_i v_i v_i^T \quad (2.57)$$

Et comme les vecteurs propres v_i de A décrivent \mathbb{R} :

$$x_0 - x^* = \sum_{i=1}^n \xi_i v_i \quad (2.58)$$

Nous introduisons ici une propriété de $P_k(A)$ du fait de sa construction :

$$P_k(A)v_i = P_k(\lambda_i)v_i \forall i = 1, 2, \dots, n \quad (2.59)$$

Ainsi, en utilisant les équations ci-dessus :

$$x_{k+1} - x^* = \sum_{i=1}^n [1 + \lambda_i P_k^*(\lambda_i)] \xi_i v_i \quad (2.60)$$

On rappelle cette propriété du calcul d'une norme selon une matrice A :

$$\|z\|_A^2 = \sum_{i=1}^n \lambda_i (v_i^T z)^2$$

Ce qui nous permet d'écrire :

$$\|x_{k+1} - x^*\|_A^2 = \min_{P_k} \sum_{i=1}^n \lambda_i [1 + \lambda_i P_k(\lambda_i)]^2 \xi_i^2 \quad (2.61)$$

Le minimum apparaît du fait que le polynôme généré est optimal au regard de la minimisation réalisé dans l'espace de Krylov (Propriété non démontrée ici).

Nous avons donc exprimé la converge de notre algorithme. Le taux de convergence du gradient conjugué est donné par :

$$\min_{P_k} \max_{i \leq i \leq n} [1 + \lambda_i P_k(\lambda_i)]^2 \quad (2.62)$$

Nous en déduisons les propriétés suivantes (non démontrées) :

- Si A possède $r < n$ valeurs propres distinctes alors le gradient conjugué va se terminer avec au plus r itérations
- Soit λ_i les valeurs propres de A. Nous avons alors :

$$\|x_{k+1} - x^*\|_A^2 \leq \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1} \right)^2 \|x_0 - x^*\|_A^2 \quad (2.63)$$

Grâce à toutes ses propriétés, on peut montrer que le gradient conjugué est bien meilleurs que les méthodes de descentes classiques. En effet, il converge bien plus vite et nous sommes sûr qu'il atteindra la solution de notre problème.

2.3.3 Algorithme du gradient conjugué

2.3.4 Quelques mots sur le gradient conjugué non-linéaire

Le but est ici de voir si l'on peut améliorer l'algorithme ci-dessus pour l'appliquer à des fonctions quadratiques non convexes voir non linéaires. Nous n'allons présenter ici que la méthode de Fletcher-Reeves. Cette méthode consiste en deux petits changements dans l'algorithme du CG.

La première modification consiste à changer le calcul du pas de descente. En effet, comme nous ne sommes plus dans le cas quadratique, il est beaucoup moins facile de trouver le pas de descente. Afin de trouver ce pas optimal, il nous faut minimiser la fonction partielle dans la direction p_k . De plus, le résidue qui était jusqu'à lors le gradient de la fonction quadratique devient ici le gradient de la fonction non-linéaire. Cette partie n'étant qu'une ouverture à l'étude non-linéaire, nous allons uniquement détailler sommairement comment choisir le nouvel α et présenter un algorithme.

Tout d'abord, voici l'algorithme de Fletcher-Reeves :

2.4 D'autres algorithmes de minimisation

Dans cette partie, nous nous intéressons aux algorithmes de Newton qui sont d'autres algorithmes permettant une nouvelle approche des problèmes de minimisation.

2.4.1 Les algorithmes de Newton

Le but d'un algorithme de Newton est d'approximer notre fonction par le développement de Taylor de celle-ci à l'ordre 2. On peut donc dès lors remarquer que nous aurons besoin de la seconde dérivée de cette fonction (ou de la Hessienne en dimension supérieure). Le développement de Taylor est :

$$f(x+h) = f(x) + \langle h, \nabla f(x) \rangle + \frac{1}{2} \langle h, H_f(x)h \rangle + o(\|h\|^2) \quad (2.64)$$

Le principe est assez simple : on se trouve au point x et nous allons choisir le point $x+h$ comme prochain point tel que ce nouveau point minimise le développement de Taylor ci-dessus. Ce développement de Taylor s'apparente à une fonction quadratique.

Dans ce type de méthode, nous choisissons notre direction de descente de la manière suivante :

$$d_k = -\nabla^2 f_k^{-1} \nabla f_k \quad (2.65)$$

Bien entendu, dans la réalité, nous ne calculons pas l'inverse de la hessienne mais nous résolvons un système linéaire. Cette méthode peut donc paraître un peu lourde.

2.4.2 Les algorithmes de Quasi-Newton

Les algorithmes de Quasi-Newton se basent sur les algorithmes de Newton. Cependant, l'amélioration vient du fait qu'ils ne nécessitent pas la dérivée seconde de la fonction que l'on cherche à minimiser. En effet, nous allons choisir une direction de descente de la manière suivante :

$$d_k = -B^{-1}\nabla f_k \quad (2.66)$$

Avec B une matrice définie positive qui est recalculée à chaque itération afin d'approximer la valeur de la hessienne. L'algorithme de Quasi-Newton que nous allons étudier ci-dessous est la méthode BFGS. Il existe plusieurs dérivées de cette méthode mais elle repose sur le même principe de base.

Chapitre 3

Comparaison logiciel des méthodes

chris je sais pas comment tu veux organiser ta partie

Chapitre 4

Applications des méthodes de descente au Machine Learning

4.1 Introduction et Motivation

Un des nouveaux domaines qui a émergé ces dernières années concerne le Machine Learning.

4.2 Application des algorithmes

Chapitre 5

Conclusion

Table des figures