République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ MOULOUD MAMMERI DE TIZI-OUZOU

FACULTÉ DES SCIENCES

DÉPARTEMENT DE MATHÉMATIQUES

Mémoire de fin d'études en vue de l'obtention du Diplôme de Master en

Mathématiques appliquées à la gestion

Thème :

Optimisation non linéaire et application

Présenté par :

OUALI Nassim

Encadré par :

Pr OUANES Mohand

Année universitaire 2017-2018

Remerciements

Je tiens, au terme du travail, à présenter mes vifs remerciements à toutes les personnes qui ont contribués, de prés ou du loin, à son bon déroulement.

Je tiens à présenter tous mes respects au Pr OUANES Mohand pour m'avoir offert l'opportunité d'effectuer ce mémoire, et aussi pour l'encadrement durant la réalisation de ce projet de fin d'étude.

Je remercie également les membres du jury qui ont accepté d'évaluer mon travail.

Mes remerciements vont aussi à tous mes professeurs, enseignants et toutes les personnes qui m'ont soutenu jusqu'au bout, et qui n'ont pas cessé de me donner des conseils très importants en signe de reconnaissance.

Dédicaces

Que ce travail témoigne de mes respects :

A mes parents :

Grâce à leurs tendres encouragements et leurs grands sacrifices, ils ont pu créer le climat affectueux et propice à la poursuite de mes études. Aucune dédicace ne pourrait exprimer mon respect, ma considération et mes profonds sentiments envers eux.

Je prie le bon Dieu de les bénir, de veiller sur eux, en espérant qu'ils seront toujours fiers de moi.

Que dieu leur accorde santé et prospérité

A mon frère**, le bijou de la famille : **Belaid (kamel)

Je lui souhaite un avenir plein de joie, de bonheur, de réussite et sérénité. Je lui exprime à travers ce travail mes sentiments de fraternité et d'amour.

Que je l'aime beaucoup

***Ma grande famille**, mon grand-père, ma grand-mère, oncles et cousins.*

A tous mes professeurs :

Leur générosité et leur soutien m'oblige de leur témoigner mon profond respect et ma loyale considération.

A tous mes amis, à qui je souhaite tout le succès dans leur vie.

Table de matières

Introduction générale

L'optimisation et plus particulièrement la programmation mathématique, vise à résoudre des problèmes où l'on cherche à déterminer parmi un grand nombre de solutions candidates, celle qui donne le meilleur résultat de la fonction objectif. Plus précisément, on cherche à trouver une solution satisfaisant un ensemble de contraintes, et qui minimise ou maximise une fonction donnée, voir [1,2,3,4,5,6,7,8,9,10,11]. L'application de la programmation mathématique est en expansion croissante et trouve beaucoup d'applications dans plusieurs domaines pratiques.

L'optimisation non linéaire s'occupe principalement des problèmes d'optimisation dont les données, i.e., les fonctions définissant ces problèmes, sont non linéaires (bien sûr), mais sont aussi différentiables autant de fois que nécessaire pour l'établissement des outils théoriques, comme les conditions d'optimalités, ou pour la bonne marche des algorithmes de résolution qui y sont introduits et analysés. Cette sous-discipline de l'optimisation, a aussi son existence liée à la communauté de chercheurs qui se sont spécialisés sur ces sujets et au type de résultats qui ont pu être obtenus.

L'objet de ce mémoire est d'implémenter une méthode de résolution de problème d'optimisation non linéaire et d'appliquer cette méthode sur le problème discrimination, c'est-à-dire décider à quelle classe appartient un échantillon, et de régression, c'est-à-dire prédire la valeur numérique d'une variable. Les SVMs (Séparateurs à Vaste Marge, en anglais Support Vector Machine) reposent sur deux idées clés : la notion de marge maximale et la notion de fonction noyau. Ces deux notions existaient depuis plusieurs années avant qu'elles ne soient mises en commun pour construire les SVMs.

Ce travail s'articule autour de trois chapitres :

Le premier chapitre se décompose en deux parties essentielles :

*1ière partie: Nous abordons les conditions d'optimalité de l'optimisation sans contraintes ainsi que la description des méthodes de résolution de ce type de problèmes d'optimisation. Parmi lesquelles on peut citer :

- Méthodes d'optimisation basées sur le gradient.
- Méthode de Newton.
- Méthodes utilisant des directions conjuguées.
- Méthode du simplexe.

* 2ième partie: cette partie comporte les conditions d'optimalité sous contraintes et les méthodes de résolution pour cette optimisation, à savoir :

- Méthodes de directions admissibles.
- Méthodes de plans sécants.
- Méthodes de pénalité.

Le deuxième chapitre rappelle, premièrement, les notions de bases des SVMs linéaires, ainsi les SVMs non linéaires (les noyaux) sont discutées.

Le dernier chapitre est consacré à l'implémentation.

Enfin, ce mémoire s'achève par une conclusion générale.

I. Optimisation non linéaire

1. Optimisation sans contraintes [13]

1.1 Conditions d'optimalité

Le problème que l'on étudie ici est celui de la recherche du *minimum* (ou du *maximum*) d'une fonction réelle f de n variables réelles $x_1, x_2, \dots, x_n$ chacune de ces variables pouvant prendre n'importe quelle valeur de $-\infty$ à $+\infty$.

De tels problèmes apparaissent fréquemment dans les applications. D'autres part, dans le cas, souvent rencontré lui aussi, où les variables $x_1, x_2, \dots, x_n$ sont astreintes à vérifier des conditions supplémentaires (du type : $g_i(x) = 0, i = 1, \dots, m$) nous verrons que l'on peut, dans certaine conditions, se ramener à la résolution de problèmes d'optimisation sans contraintes.

Soit $f: \mathbb{R}^n \to \mathbb{R}$ qui a tout $x \in \mathbb{R}^n$, $x = (x_1, x_2, \dots, x_n)^T$, associe la valeur réelle :

$$f(x) = f(x_1, x_2, \dots, x_n)$$

On cherche à résoudre :

$$\begin{cases} Min\ f(x) \\ x \in \mathbb{R}^n \end{cases}$$

Il s'agit donc de déterminer un point x^* de $\mathbb{R}^n$ tel que :

$$\forall x \in \mathbb{R}: \qquad f(x^*) \leq f(x) \tag{1}$$

C'est-à-dire un *minimum global* de f sur $\mathbb{R}^n$.

Une condition suffisante pour l'existence de x^* vérifiant (1) est que f soit continue (ou semi continue inférieurement) et possède la propriété de croissance à l'infini : $f(x) \to +\infty$ lorsque $\|x\| \to +\infty$.

Lorsque l'inégalité stricte $f(x^*) < f(x)$ est vérifiée $\forall x \in \mathbb{R}^n$, le *minimum global* est unique, mais l'unicité n'est pas toujours vérifiée.-

Pour beaucoup de problèmes d'optimisation sans contrainte, en l'absence de propriétés particulières (telles que la convexité ou la quasi-convexité), les principales méthodes de résolution connues ne permettent pas la détermination d'un minimum global : il faut alors se contenter d'*optimums locaux*, c'est-à-dire de points qui vérifient (1) seulement dans un voisinage x^*. Nous allons voir, en ce qui suit, comment de tels points peuvent être caractérisés.

Nous commencerons alors par nous placer dans l'hypothèse où la fonction est continue et deux fois différentiable, et nous discuterons successivement des

conditions d'optimalité puis les principales méthodes numériques pour la résolution de problèmes d'optimisation sans contraintes.

1.1.1 Conditions nécessaires d'optimalité locale

Supposons que f est continue et a des dérivées partielles premières $\partial \mathrm{f}/\,\partial \mathrm{x_i}$; $i = 1, \dots, n$; et secondes $\partial^2 \mathrm{f}/\,\partial \mathrm{x_i}\,\partial \mathrm{x_j}$; $i, j = 1, \dots, n$; continues pour tout $x \in \mathbb{R}^n$.

Théorème 1 :

Une condition nécessaire pour que x^* soit un minimum local de f est :

(a) $\nabla f(x^*) = 0$ (stationarité)

(b) $\nabla^2 f(x^*) = [\partial^2 f/\partial x_i \partial x_j(x^*)]$ est une matrice semi-définie positive.

La condition (a) est appelée *condition nécessaire du premier ordre* et les conditions (a) et (b) sont appelées *conditions nécessaires du second ordre*.

Démonstration : Soit x^* un minimum local de f.

Comme f est deux fois continument différentiable, le développement de Taylor à l'ordre 2 au voisinage de x^* donne:

$f(x) = f(x^*) + \nabla f^T(x^*)(x - x^*) + \frac{1}{2}(x - x^*)^T \nabla^2 f(x^*)(x - x^*) + \|x - x^*\|^2 \mathcal{E}(x - x^*)$

Avec $\mathcal{E}(x - x^*) \to 0$ quand $x \to x^*$.

(a) Si $\nabla f(x^*) \neq 0$ alors en choisissant $x = x^* - \theta \nabla f(x^*)$, pour $\theta > 0$ suffisamment petit, alors: $f(x) < f(x^*)$ ce qui contredirait le fait que x^* est un minimum local. Donc la condition (a) est bien nécessaire, et on peut l'écrire :

$$f(x) = f(x^*) + \frac{1}{2}(x - x^*)^T \nabla^2 f(x^*)(x - x^*) + \|x - x^*\|^2 \mathcal{E}(x - x^*)$$

(b) Si la matrice $\nabla^2 f(x^*)$ n'est pas semi-définie positive, c'est qu'il existe un vecteur $\mathrm{d} \in \mathbb{R}^n$ $(d \neq 0)$ tel que: $d^T \nabla^2 f(x^*) d < 0$. En choisissant alors $x = x^* + \theta d$, pour $\theta > 0$ suffisamment petit on aurait $f(x) > f(x^*)$ ce qui contredirait encore l'optimalité locale de x^*. La condition (b) est donc bien nécessaire aussi.

Remarque :

Un point x^* qui vérifié la condition (a) est appelé un *point stationnaire (critique)*.

Ces conditions sont nécessaires mais elles ne sont pas suffisantes pour garantir un minimum local.

1.1.2 Conditions suffisantes d'optimalité locale

Théorème 2 :

Sous les mêmes hypothèses qu'avant, une condition suffisante pour que x^* soit un optimum local de f sur $\mathbb{R}^n$ est :

(a) $\nabla f(x^*) = 0$ (stationarité)

(b) $\nabla^2 f(x^*)$ est une matrice définie positive

Démonstration : Considérons un point x^* satisfaisant les deux conditions (a) et (b) du théorème. Le développement de Taylor de f au voisinage de x^* s'écrit alors :

$$f(x) = f(x^*) + \frac{1}{2}(x - x^*)^T \nabla^2 f(x^*)(x - x^*) + \|x - x^*\|^2 \mathcal{E}(x - x^*)$$

avec $\mathcal{E}(x - x^*) \to 0$ quand $x \to x^*$
Pour toute direction de déplacement $\mathrm{d} \in \mathbb{R}^n$ $(\|d\| = 1)$ on a alors

$$f(x^* + \theta d) = f(x^*) + \frac{\theta^2}{2} d^T \nabla^2 f(x^*) d + \theta^2 d^2 \mathcal{E}(\theta d)$$

où $\mathcal{E}(\theta d) \to 0$ quand $\theta \to 0$.

En vertu de la condition (b) on a $d^T \nabla^2 f(x^*) d > 0$ et par suite, pour θ suffisament petit, on aura : $f(x^* + \theta d) > f(x^*)$

Ce qui montre que x^* est bien un minimum local de f.

La condition (b) revient à supposer que f est *strictement convexe* dans un voisinage de x^*.

1.1.3 Conditions suffisantes d'optimalité globale

Dans le cas d'une fonction convexe f définie sur $\mathbb{R}^n$, on a la condition nécessaire et suffisante pour que x^* soit un minimum global de f est donnée dans le théorème suivant:

Théorème 3 :

Soit une fonction continue $f: \mathbb{R}^n \to \mathbb{R}$ et $x^* \in \mathbb{R}^n$un minimum local de f.

- Si f est une fonction convexe, alors x^* est un minimum global de f.
- Si de plus f est strictement convexe, x^* est l'unique minimum global de f.

Autrement dit, dans le cas convexe, la stationnarité à elle seule constitue une condition nécessaire et suffisante d'optimalité globale.

Démonstration : Supposons par l'absurde qu'il existe un autre minimum local $x^+ \neq x^*$, tel que $f(x^+) < f(x^*)$. Par la convexité de f, nous avons : $\forall \lambda \in [0,1]$;

$$f(\lambda x^* + (1-\lambda)x^+) \leq \lambda f(x^*) + (1-\lambda) f(x^+) \leq \lambda f(x^*) + (1-\lambda) f(x^*) < f(x^*) \qquad (2)$$

La définition d'un minimum local est contredite.

Considérons maintenant une fonction strictement convexe, et supposons que x^* et y^* soient deux minima globaux distincts, et donc $\mathrm{x}^* \neq \mathrm{y}^*$ et $(x^*) = f(y^*)$.

En fonction de la définition de convexité stricte, nous avons :

$$\forall \lambda \in [0,1]; \qquad f(\lambda x^* + (1-\lambda) y^+) < \lambda f(x^*) + (1-\lambda) f(y^*) = f(y^*)$$

ce qui contredit que x^* et y^* sont des minimas globaux.

1.1.4 Conditions suffisantes pour les problèmes quadratiques

Parmi les fonctions non linéaires, les fonctions quadratiques jouent un rôle important dans les algorithmes d'optimisation.

Définition : Fonction quadratique :

Une fonction $f : \mathbb{R}^n \to \mathbb{R}$ sera dite quadratique si elle peut s'écrire sous forme :

$$f(x) = \frac{1}{2} x^T A x + b^T x + c \qquad (3)$$

où A est une matrice symétrique $n \times n$, $b \in \mathbb{R}^n$ et $c \in \mathbb{R}$. Nous avons alors

$$\nabla f(x) = Ax + b$$

Et

$$\nabla^2 f(x) = A$$

Théorème 4 :

Considérons le problème

$$\min_{x\in\mathbb{R}^n} f(x) = x^T Ax + b^T x + c \qquad (4)$$

où A est une matrice symétrique $n \times n$, $b \in \mathbb{R}^n$ et $c \in \mathbb{R}$.

4. *Si A n'est pas semi-définie positive, alors le problème (4) ne possède pas de solution, c'est-à-dire qu'il n'existe aucun $x \in \mathbb{R}^n$ qui soit un minimum local de (3).*
5. *Si A est définie positive, alors*

$$x^* = -A^{-1}b \qquad (5)$$

est l'unique minimum global de (3)

Démonstration :

Nous avons $\nabla f(x) = Ax + b$ et $\nabla^2 f(x) = A$

1. Supposons par l'absurde qu'il existe x^* minimum local de (2.3). D'après (b) du théorème 1, est semi-définie positive, ce qui contredit l'hypothèse.
2. Comme A est définie positive, le point x^* dans (5) est bien défini car:

$$\nabla f(x^*) = Ax^* + b$$

On remplace x^* par sa valeur, on aura

$$\nabla f(x^*) = -AA^{-1}b + b = 0$$

Les conditions suffisantes d'optimalité du théorème (2) sont vérifiées et x^* est un minimum local de f. De plus, f est convexe. D'après le théorème (3), x^* est l'unique minimum global.

1.2 Méthodes d'optimisation sans contrainte

1.2.1 Méthodes de descente basées sur le gradient [13]

Les méthodes basées sur le gradient de la fonction objectif sont des procédures parmi les plus fondamentales pour minimiser une fonction différentiable de $\mathbb{R}^n$ dans $\mathbb{R}$. Comme la plupart des autres méthodes développées pour ce problème, elles reposent sur la propriété dite de *descente itérative*. Rappelons qu'un algorithme itératif part d'un vecteur $x^0 \in \mathbb{R}^n$et génère une suite de vecteur $x^1, x^2, \ldots$ de $\mathbb{R}^n$, la propriété de descente itérative impliquant que le coût des vecteurs ainsi générés décroisse à chaque itération :

$$f(x^{k+1}) < f(x^k) \qquad \forall k \in N$$

Les méthodes basées sur le gradient appartiennent à une importante classe d'algorithmes, où les vecteurs sont générés de la manière suivante :

$$x^{k+1} = x^k + a_k^* d^k \qquad a_k > 0 \quad \text{est le pas}$$

Il paraît approprié de donner ici la définition de la notion de direction de descente :

Définition : direction de descente :

Soient $f: \mathbb{R}^n \to \mathbb{R}$. Un vecteur d est appelé une *direction de descente* de f en x si $\exists a > 0$ tel que $f(x + ad) < f(x)$, $\forall a \in [0,1]$ $\quad a$ *est le pas*

Ainsi, afin d'assurer que la propriété de descente itérative soit garantie par l'algorithme, la direction d^k choisie, dans l'équation ci-dessus, doit être une direction de descente. Toute direction d^k telle que $\nabla f(x^k) \cdot d^k < 0$ est une direction de

descente de f en x^k. Les algorithmes de ce type sont appelés *méthodes du gradient* en raison de cette relation entre la direction d^k et le gradient $\nabla f(x^k)$.

Il n'y a pas de nom universellement reconnu et utilisé pour ces méthodes. Certains auteurs réservent l'appellation "*méthode du gradient*" au cas où $d^k = -\nabla f(x^k)$ (celle-ci est également parfois appelée *méthode optimale du gradient*), alors que d'autre l'emploient afin de désigner l'ensemble des algorithmes de ce type, pour lesquels de nombreuses stratégies de choix de d^k sont possibles.

Venons-en maintenant à la méthode du gradient : il s'agit précisément du cas particulier ou $d^k = -\nabla f(x^k)$; c'est pourquoi dorénavant, dans le présent document, l'appellation "*méthode du gradient*" sera utilisée uniquement en référence à ce cas précis où d^k est choisie et où a^k est déterminé suivant la politique dite de minimisation (voir ci-dessous). C'est une propriété intéressante de la direction $-\nabla f(x^k)$ qui nous conduit à ce choix : parmi toutes les directions $d \in \mathbb{R}^n$normalisées, il s'agit de celle qui minimise la dérivée directionnelle $\nabla f(x^k).d$ de f en x^k, comme nous allons le voir dans la proposition ci-après.

Le problème de recherche de cette direction consiste à trouver la direction d qui minimise $\nabla f(x).d$ sous contrainte $\|d\| = 1$. La proposition ci-dessous stipule que la direction $d = -\nabla f(x)/\|\nabla f(x)\|$ est la solution optimale de ce problème.

Proposition :

Soient $f: \mathbb{R}^n \to \mathbb{R}$ continuellement différentiable et $x \in \mathbb{R}^n$. Supposons que $\nabla f(x) \neq 0$. Alors le problème consistant à minimiser $f'(x, d)$ sous contrainte $|d| = 1$ a pour solution optimale $d^* = -\nabla f(x)/|\nabla f(x)|$.

La direction de descente choisie sera, à chaque itération, $d^k = -\nabla f(x^k)/|\nabla f(x^k)|$. Les points sont ainsi successivement générés par la méthode du gradient de la manière suivante :

$$x^{k+1} = x^k - a^k \nabla f(x^k), \qquad a^k > 0$$

Le point x^0 peut être choisi arbitrairement. Il importe désormais de choisir a^k d'une manière aussi convenable que possible. Tout comme pour d^k, diverses stratégies peuvent être employées pour ce choix. Celle que nous adoptons ici est généralement désignée sous l'appellation de *règle de minimisation*. Elle consiste à choisir, à chaque itération, a^k comme étant la solution optimale du problème de minimisation monodimensionnelle de f le long de la demi-droite définie par le point x^k et la direction $-\nabla f(x^k)$. Un tel sous-problème peut être résolu de diverses manières, par exemple par la méthode de Newton.

Remarquons que la méthode s'arrête lorsque $\nabla f(x^k) = 0$ car dans ce cas $x^{k+1} = x^k$. Nous allons aborder dans la section suivante les questions plus particulièrement relatives à la convergence de cette méthode.

Convergence de la méthode :

Idéalement, nous souhaiterions pouvoir générer, à l'aide de la méthode du gradient, une séquence $\{x^k\}$ convergeant vers un minimum global de f. Cependant, c'est bien sûr trop demander à une telle méthode, du moins si f n'est pas convexe (en raison de la présence d'extrêma locaux qui ne sont pas globaux). La méthode du gradient est guidée localement selon la forme de f dans la région correspondante au point x^k, et peut ainsi être attirée par tout type de minimum, qu'il soit local ou global. Remarquons que si, pour une quelconque raison, la méthode est démarrée depuis ou rejoint un maximum ou un point stationnaire, elle se termine en ce point. Ainsi, si

f n'est pas convexe, nous pouvons, au mieux, attendre de la méthode du gradient qu'elle converge vers un point stationnaire.

Théorème : **[5]**

Soit $\{x^k\}$ une séquence générée par la méthode du gradient. Alors tout point limite de $\{x^k\}$ est un point stationnaire.

Il peut arriver que la méthode du gradient converge de manière finie, mais ce n'est en général pas le cas. Il est donc nécessaire d'utiliser un critère permettant d'arrêter l'exécution lorsque x^k est suffisamment proche d'un point stationnaire, par exemple $|\nabla f(x)| < \mathcal{E}$, où $\mathcal{E}$ est un scalaire positif arbitrairement choisi. A priori, la valeur que nous devons fixer pour $\mathcal{E}$ dépend du problème considéré.

L'inconvénient majeur de la méthode du gradient survient lorsque les surfaces de coût égal de f sont "allongées", et x^k est tel que la direction du gradient y est presque orthogonale à la direction menant au minimum. Celle-ci adopte alors le comportement bien connu du "zigzag" et progresse extrêmement lentement, comme nous aurons l'occasion de le constater lors de l'application pratique de la méthode.

Il existe des moyens de surmonter ces difficultés en choisissant la direction d^k un peu différemment. La direction donnée par le gradient peut être corrigée en la multipliant par une matrice ou en y ajoutant un vecteur approprié : la performance peut être améliorée en se déplaçant dans la direction $d^k = -D\nabla f(x^k)$ou $d^k = \nabla f(x^k) + v$, où D et v sont convenablement choisis.

1.2.2 Méthodes de Newton [5]

Alors que la méthode du gradient utilise une approximation linéaire pour trouver une direction de mouvement, l'idée de la méthode itérative de Newton est de minimiser à chaque itération l'approximation quadratique de f au point courant x^k et donnée par le développement de Taylor d'ordre 2 :

$$q^k(x) = f(x^k) + \nabla f(x^k) \cdot (x - x^k) + \frac{1}{2}(x - x^k) \cdot \nabla^2 f(x^k)(x - x^k)$$

Une condition nécessaire pour que le minimum de $q^k(x)$ soit atteint est $\nabla q^k(x) = 0$ soit :

$$\nabla f(x^k) + \nabla^2 f(x^k)(x - x^k) = 0$$

Le vecteur généré à l'itération $k + 1$ est le vecteur minimisant $q^k(x)$, c'est-à-dire le vecteur satisfaisant l'équation précédente, soit

$$x^{k+1} = x^k - (\nabla^2 f(x^k))^{-1} \nabla f(x^k)$$

La méthode nécessitant l'évaluation de la matrice hessienne de f, elle ne peut être utilisée que si f est deux fois continument différentiable (la méthode de Newton requiert même l'évaluation de l'inverse de cette matrice, ce qui est coûteux en termes de calculs). On peut remarquer que la méthode s'arrête également lorsque $\nabla f(x^k) = 0$, car il s'ensuit que $x^{k+1} = x^k$. Si, en plus, $\nabla^2 f(x^*)$ est définie positive, alors la condition suffisante donnée au théorème 2.2.3 est satisfaite, impliquant que x^k soit un minimum local.

Notons que la méthode de Newton converge en une seule itération si f est quadratique.

Convergence de la méthode :

La méthode de Newton décrite ci-dessus présente plusieurs inconvénients :

1. L'inverse de la matrice hessienne $(\nabla^2 f(x^k))^{-1}$ peut ne pas exister, auquel cas la méthode échoue. Cela intervient typiquement lorsque la méthode atteint une région où f est linéaire (ses secondes dérivées partielles valent zéro).
2. La méthode de Newton n'est pas une méthode de descente : il est possible que $f(x^{k+1})$ soit supérieur à $f(x^k)$.
3. Elle est attirée aussi bien par les minima que par les maxima locaux (cette propriété est liée à la précédente). En effet, la méthode, à chaque itération, recherche uniquement un point tel que le gradient de l'approximation quadratique soit égal au vecteur nul, que ce point soit un maximum, un minimum ou un point stationnaire.

La méthode ne converge donc pas en général, notamment si elle est démarrée loin d'un minimum local, pour les première et troisième raisons. Cependant, elle converge sous certaines restrictions : si elle est exécutée à partir d'un point suffisamment proche d'un minimum local et que $\nabla^2 f(x^k)$ n'est pas singulière, alors la méthode de Newton convergera vers ce minimum (mais pas de manière finie, de sorte qu'une condition d'arrêt est requise de façon analogue à la méthode du gradient) et la convergence est quadratique.

1.2.3 Méthodes utilisant des directions conjuguées [13]

Ces méthode sont basées sur l'important concept de la conjugaison et ont été développées afin de résoudre le problème quadratique

$$\text{minimiser } f(x) = \frac{1}{2} x \cdot Qx + b \cdot x + c$$
$$\text{avec } x \in \mathbb{R}^n$$

où $Q \in \mathbb{R}^{n \times n}$ est symétrique et définie positive, $b \in \mathbb{R}^n$ et $c \in \mathbb{R}$. Les méthodes de direction conjuguées peuvent résoudre les problèmes de cette forme en au plus n itérations et, contrairement aux méthodes présentées jusqu'à présent, elles n'utilisent pas de dérivées, sauf dans le cas particulier de la méthode du gradient conjugué.

Définition :

Soient Q une matrice $n \times n$ symétrique et définie positive et un ensemble de directions non nulles $d^1, d^2, \dots, d^k$. Ces directions sont dites *Q-conjuguées* si

$$d^i \cdot Qd^j = 0, \quad \forall i, j \text{ tels que } i \neq j$$

Si $d^1, d^2, \dots, d^k$ sont *Q*-conjuguées, alors elles sont linéairement indépendantes. Etant donné un ensemble de *n* directions *Q*-conjuguées $d^0, d^1, \dots, d^{n-1}$, la méthode de directions conjuguées correspondante est donnée par

$$x^{k+1} = x^k + a^k d^k, \qquad k = 0, \dots, n-1$$

où x^0 est un vecteur de départ choisi arbitrairement et où les a^k sont obtenus par minimisation monodimensionnelle le long de d^k.

Le principal résultat concernant les méthodes utilisant des directions conjuguées est qu'à chaque itération k, la méthode minimise f sur le sous-espace généré par les k premières directions *Q*-conjuguées utilisées par l'algorithme. A la *n*ième itération au plus tard, ce sous-espace inclura alors le minimum global de f, grâce à la propriété d'indépendance linéaire des directions *Q*-conjuguées qui assure qu'à l'itération n, l'espace vectoriel généré par les *n* directions *Q*-conjuguées ne sera autre que $\mathbb{R}^n$.

Comment construire les directions Q-conjuguées :

Des directions *Q*-conjuguées $d^0, \dots, d^k$ peuvent être générées à partir d'un ensemble de vecteurs linéairement indépendants $\mathcal{E}^0, \dots, \mathcal{E}^k$ en utilisant la procédure dite *de Gram-Schmidt*, de telle sorte que pour tout i entre 0 et k, le sous-espace généré par $d^0, \dots, d^i$ soit égale au sous-espace généré par $\mathcal{E}^0, \dots, \mathcal{E}^i$. Cette procédure fonctionne de la manière suivante. Elle commence par choisir $d^0 = \mathcal{E}^0$. Supposons maintenant que les directions $d^0, \dots, d^i$ ont été construites $i < k$, satisfaisant la propriété précédente. Alors d^{i+1} est construite comme suit :

$$d^{i+1} = \mathcal{E}^{i+1} + \sum_{m=0}^{i} c^{(i+1)m} d^m$$

Nous pouvons noter que si d^{i+1} est construite d'une telle manière, elle est effectivement linéairement indépendante de $d^0, \dots, d^i$. En effet, le sous-espace généré par les directions $d^0, \dots, d^i$ est le même que le sous-espace généré par les directions $\mathcal{E}^0, \dots, \mathcal{E}^{i+1}$ et $\mathcal{E}^{i+1}$ est linéairement indépendante de $\mathcal{E}^0, \dots, \mathcal{E}^i \cdot \mathcal{E}^{i+1}$ ne fait donc pas partie du sous-espace généré par les combinaisons linéaires de la forme $\sum_{m=0}^{i} c^{(i+1)m} d^m$, de sorte que d^{i+1} n'en fait pas partie non plus et est donc linéairement indépendante des $d^0, \dots, d^i$.

Les coefficients $c^{(i+1)m}$, sont choisis de manière à assurer la *Q*-conjugaison des $d^0, \dots, d^{i+1}$, c'est-à-dire de telle sorte que pour tout $j = 0, \dots, i$, nous ayons $d^{i+1} \cdot Qd^j = 0$

La méthode du gradient conjugué : [13]

La méthode du gradient conjugué est obtenue en appliquant la procédure de Gram-Schmidt aux gradients $\nabla f(x^0), \dots, \nabla f(x^{n-1})$, c'est-à-dire en posant

$$\mathcal{E}^0 = -\nabla f(x^0), \dots, \mathcal{E}^{n-1} = -\nabla f(x^{n-1})$$

En outre, nous avons que

$$\nabla f(x) = Qx + b$$

et

$$\nabla^2 f(x) = Q$$

Notons que la méthode se termine si $\nabla f(x^k) = 0$. La particularité intéressante de la méthode du gradient conjugué est que le membre de droite de l'équation donnant la valeur de d^{k+1} dans la procédure de Gram-Schmidt peut être grandement simplifié. En particulier, nous pouvons vérifier que seul un des termes de la somme est non nul, de sorte que cette méthode, implémentée, se révèle ainsi plus rapide que la méthode décrite précédemment qui applique la procédure de Gram-Schmidt à des vecteurs linéairement indépendants quelconques.

Algorithme de la méthode du gradient conjugué

1. **Initialisation**

   Fixer $\mathcal{E} > 0$, choisir $x^0 \in \mathbb{R}^n$ poser $\nabla f(x^0) = Ax^0 - b$ et $d^0 = -\nabla f(x^0)$

2. **Itération** $k = 0,1, \dots$

   Calculer $a_k = -\frac{(d^k)^T \nabla f(x^k)}{(d^k)^T A d^k}$

   Calculer $x^{k+1} = x^k + a_k d^k$

   Calculer $\nabla f(x^{k+1}) = f(x^{k+1}) + a_k A d^k$

Calculer $\beta^{k+1} = \frac{\|\nabla f(x^{k+1})\|^2}{\|\nabla f(x^k)\|^2}$

Calculer $d^{k+1} = -\nabla f(x^{k+1}) + \beta^{k+1} d^k$

3. Critère d'arrêt

Si $\|x^{k+1} - x^k\| < \mathcal{E}$, STOP, x^{k+1} est la solution optimale
Sinon, on pose $k = k + 1$ et on retourne à 2.

Convergence des méthodes de directions conjuguées :

Théorème : **[13]**

Soit x^n le point obtenu à la *n*ième itération d'une méthode de directions conjuguées, pour x^0 quelconque. Alors x^n est un minimum global de f sur $\mathbb{R}^n$.
La méthode converge donc de manière finie, pour peu que les hypothèses précédentes soient respectées.

Il convient de mentionner qu'il est également possible d'employer les méthodes de directions conjuguées pour résoudre des problèmes non quadratiques. Cependant, il faut pour cela recourir à des stratégies permettant de surmonter la "perte" de conjugaison des directions générées due à la présence de termes non quadratiques dans la fonction objectif.

1.2.4 Méthode du simplexe [13]

C'est un euphémisme que de dire que la méthode du simplexe (à ne pas confondre avec la méthode du simplexe pour la programmation linéaire), également appelée méthode de Nelder et Mead, fonctionne différemment des autres méthodes décrites jusqu'ici. En effet, elle ne consiste pas à déterminer, à chaque itération, une direction le long de laquelle se déplacer depuis x^k pour obtenir x^{k+1}. Son principe est de maintenir un ensemble de $n+1$ points, appelé *simplexe*. A chaque itération, l'algorithme remplace le point de coût maximum x_{max} dans le simplexe par un autre dont le coût est inférieur. Ce nouveau point est déterminé par l'algorithme d'une manière très "géométrique", suivant les étapes de réflexion, d'expansion ou de contraction qui sont décrites ci-dessous.

Décrivons précisément l'algorithme. Avant le début de l'exécution, le choix d'un simplexe initial doit être opéré. Soient $x^0, x^1, \dots, x^n$ les $n+1$ points du simplexe courant. Soient x_{min} et x_{max} les points de coût respectivement le moins et le plus élevé, c'est-à-dire :

$$f(x_{min}) = \min_{i=0,\dots,n} f(x^i)$$

et

$$f(x_{max}) = \max_{i=0,\dots,n} f(x^i)$$

Soit $\bar{x}$ défini de la manière suivante (nous pourrions le qualifier de "centre" du polyèdre formé par les points du simplexe à l'exception de x_{max}) :

L'itération consiste à remplacer le pire des points x_{max} par un point de coût inférieur ; pour cela, le point dit de réflexion x_{ref} est calculé, situé sur la droite passant par x_{max} et $\bar{x}$ et symétrique à x_{max} par rapport à $\bar{x}$:

Alors, un nouveau point x_{new} amené à remplacer x_{max} dans le simplexe est calculé, en fonction des coûts de x_{ref} et des points du simplexe autres que x_{max}. En fonction de cela, x_{new} pourra être déterminé selon trois phases différentes, décrites ci-dessous :

1. Si $f(x_{min}) > f(x_{ref})$, l'expansion est effectuée.
   Le point x_{exp} est calculé :
   $$x_{exp} = 2x_{ref} - \bar{x}$$
   Et x_{new} est défini comme suit :
   $$x_{new} = \begin{cases} x_{exp} & si \quad f(x_{exp}) < f(x_{ref}) \\ x_{ref} & sinon. \end{cases}$$
2. Si $max_{x^i \neq x_{max}} f(x^i) > f(x_{ref}) \geq f(x_{min})$, la *réflexion* est effectuée.
   x_{new} est simplement défini étant égal à x_{ref}
3. Si $f(x_{ref}) \geq max_{x^i \neq x_{max}} f(x^i)$, la *contraction* est effectuée.
   x_{new} est calculé comme suit :
   $$x_{new} = \begin{cases} \frac{1}{2}(x_{max+x}) & si \quad f(x_{exp}) < f(x_{ref}) \\ \frac{1}{2}(x_{ref+x}) & sinon. \end{cases}$$

Dans les trois cas, le nouveau simplexe est formé en remplaçant x_{max} par x_{new}.
Il n'existe pas de résultat connu quant à la convergence de cette méthode.

2. Optimisation sous contraintes [5]

2.1 Conditions d'optimalité

Nous allons maintenant discuter des conditions d'optimalité pour le problème de minimisation de $f(x)$ sous contrainte $x \in X \subset \mathbb{R}^n$, où X est défini par une collection de m inégalités et de *r* égalités :

minimiser $f(x)$
sous contraintes $g_1(x) \leq 0, \dots, g_m(x) \leq 0$
$h_1(x) = 0, \dots, h_r(x) = 0$
$x \in \mathbb{R}^n$

Donnons tout d'abord la définition de la notion de direction admissible, puis nous énoncerons un théorème qui établira une première condition nécessaire (tirée d'une propriété "géométrique") d'optimalité pour un tel problème.

Définition : direction admissible :
Soient un ensemble $X \subset \mathbb{R}^n$, avec $X \neq \emptyset$, et $x^* \in X$. Une *direction admissible* de X en x^* est un vecteur d tel que :
$d \neq 0$ et $x^* + ad \in X$, $\quad \forall a \in [0, a_{max}]$ pour un certain $a_{max} > 0$

Les conditions d'optimalité énoncées plus bas s'obtiennent suite au constat suivant : si x^* est un minimum local de f sur X, alors il ne peut y avoir aucune direction de descente dans l'ensemble des directions admissibles de X en x^* (cela peut être démontré et fournir une première condition nécessaire d'optimalité).

La condition telle que nous venons de l'énoncer ne présente malheureusement que peu d'intérêt en vue d'un usage en pratique, car si l'ensemble des directions de

descente de f en x^*, disons $D = \{d|\nabla f(x^*) \cdot d < 0\}$, peut être exprimé en fonction du gradient de la fonction objectif, ce n'est pas nécessairement le cas de l'ensemble de toutes les directions admissibles de X en x^*, $A = \{d|d \neq 0\ et\ x^* + ad \in X, \forall a \in [0, a_{max}]$ pour un certain $a_{max} > 0\}$. Il serait avantageux de trouver un moyen de la convertir en une condition plus pratiquement utilisable mettant en jeu des équations ou des inéquations.

La stratégie suivante peut être utilisée à cette fin : des sous-ensembles de A, disons G et H, peuvent être définis en fonction respectivement des gradients g_i actives en x^* et des h_i. Il découle ensuite du fait que $A \cap D = \emptyset$ soit une condition nécessaire d'optimalité le fait que $G \cap H \cap D = \emptyset$ le soit également.

2.1.1 Conditions de Lagrange

On s'intéresse ici au problème :

		Minimiser $f(x)$
(P)	avec	$g_i(x) = 0 \quad (1 \leq i \leq m)$
	et	$x \in \mathbb{R}^n$

où les fonctions f et g_i $(1 \leq i \leq m)$ sont de classe C^1. La condition de Lagrange, que donne le théorème suivant, fournit une condition nécessaire pour qu'un élément de $\mathbb{R}^n$ soit un minimum local de (P).

Théorème:
Soit x^* un minimum local du problème. Alors il existe des réels $\lambda_1, \lambda_2, ..., \lambda_m$ tels que :

$$\nabla f(x^*) = \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*)$$

Preuve:
Notons E le sous-espace de $\mathbb{R}^n$ engendré par les vecteurs $\nabla g_i(x^*)$ et $E^\perp$ le sous-espace orthogonal à E. On a :

$$\nabla f(x^*) = y + z \quad \text{avec} \quad y \in E \quad \text{et} \quad z \in E^\perp$$

Pour $1 \leq i \leq m$, $z^t.\nabla g_i(x^*) = 0$ puisque z appartient à $E^\perp$. Par conséquent, z est une direction admissible ; d'après le théorème du paragraphe précédent, il vient : $z^t.\nabla f(x^*) \geq 0$. De plus, pour $1 \leq i \leq m$, $(-z)^t.\nabla g_i(x^*) = 0$; on a donc aussi $(-z)^t.\nabla f(x^*) = 0$. D'où :

$$z^t.\nabla f(x^*) = 0$$

Or

$$z^t.\nabla f(x^*) = z^t.y + z^t.z = z^t.z = \|z\|^2$$

Par conséquent $\|z\|^2 = 0$ et donc $z = 0$. D'où le théorème.

La condition de Lagrange n'est généralement pas suffisante. Elle l'est cependant dans le cas suivant, ce que nous ne prouvons pas.

Théorème:
La condition de Lagrange est suffisante lorsque f est convexe dans un ouvert contenant X et que les g_i $(1 \leq i \leq m)$ sont linéaires.

2.1.2 Conditions de Karush-Kuhn-Tucker

On s'intéresse au problème suivant :

$$(P) \quad \text{avec} \quad \begin{cases} \text{Minimiser } f(x) \\ \begin{cases} g_i(x) = 0 \ pour \ (1 \le i \le m) \\ h_j(x) = 0 \ pour \ (1 \le j \le p) \end{cases} \\ x \in \mathbb{R}^n \end{cases} \quad \text{et}$$

où les fonctions f, $g_i(x) = 0\ (1 \le i \le m)$ et $h_j(x) = 0\ (1 \le j \le p)$ sont supposées de classe C^1. La condition suivante, appelée condition de Karush-Kuhn-Tucker, généralise la condition de Lagrange :

Théorème:
Soit x^* un minimum local du problème. Alors il existe :

- m nombres réels $\lambda_1, \lambda_2, \ldots, \lambda_m$
- p nombre réels $\mu_1, \mu_2, \ldots, \mu_p$

Tels que :

1) $\nabla f(x^*) = \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) \sum_{j=1}^{p} \mu_j \nabla h_j(x^*)$
2) Si $h_j(x^*) \neq 0$, alors $\mu_j = 0$ (dans la seconde somme, seuls les gradients des contraintes saturées en x^* peuvent figurer).

Comme la condition de Lagrange, la condition de Karush-Kuhn-Tucker n'est généralement pas suffisante. Elle l'est cependant dans le cas suivant. Nous ne prouvons pas le théorème précédent, ni le suivant.

Théorème:
La condition de Karush-Kuhn-Tucker est suffisante lorsque simultanément f est convexe dans un ouvert contenant X, les $g_i\ (1 \le i \le m)$ sont linéaires et les $h_j (1 \le j \le p)$ sont concaves dans un ouvert contenant X.

2.2 Méthodes d'optimisation sous contraintes

Nous considérons ici le problème

$$\begin{aligned} &\text{minimiser } f(x) \\ &\text{sous contrainte } x \in X \end{aligned}$$

où X est défini par une collection d'inégalités $g_i(x) \le 0$; $i = 1, \ldots, n$ et d'égalités $h_j(x) = 0$; $j = 1, \ldots, m$

Nous décrivons les principales méthodes itératives implémentées pour ce problème. Il s'agira des classes de méthodes suivantes :

1. Les méthodes de directions admissibles
2. Les méthodes de plans sécants
3. Les méthodes de pénalité
4. Les méthodes de décomposition

Les champs d'application de ces méthodes (soit les hypothèses sous lesquelles elles s'appliquent) seront notamment mentionnés et elles seront analysées du point de vue de leur convergence. Après les avoir utilisées en pratique, nous comparerons

leurs performances. Enfin, des stratégies destinées à fournir aux algorithmes un point initial admissible en partant d'un point quelconque de $\mathbb{R}^n$ seront discutées.

Remarque : toute contrainte sous forme d'égalité peut être exprimée de manière équivalente par deux inégalités :

$$h_j(x) = 0$$

peut être exprimé sous la forme

$$h_j(x) \leq 0$$
$$-h_j(x) \leq 0$$

Au travers des sections suivantes, nous admettrons donc que X n'est défini que par une collection de m inégalités $g_i(x) \leq 0$, puisque cela n'induit aucune perte de généralité. Nous agirons de la sorte pour toutes les méthodes sauf la méthode de barrière et la méthode de Zoutendijk pour les contraintes non linéaires, qui, nous verrons pourquoi, ne sont pas en mesure de traiter de contraintes sous forme d'égalités.

2.2.1 Méthodes de directions admissibles [5]

Cette classe de méthodes résout un problème de minimisation non linéaire en se déplaçant d'un point de X vers un autre de ses points au coût inférieur. Elles fonctionnent selon le principe suivant : étant donné un élément x^k de X, une direction d^kest générée telle que pour un $a^k > 0$ et suffisamment petit, les propriétés suivantes sont assurées :

1. $x^k + a^k d^k$ appartient toujours à X.
2. $f(x^k + a^k d^k)$ est inférieur à $f(x^k)$

Une fois d^k déterminée, a^k s'obtient par minimisation monodimensionnelle pour que le déplacement dans la direction d^k soit optimal, mais cette fois-ci il est nécessaire d'imposer une borne supérieure sur la valeur de a^k afin de ne pas sortir de X. Cela définit le nouveau point x^{k+1} et le processus est recommencé.

Il existe plusieurs stratégies pour la résolution du sous-problème consistant à déterminer d^k Comme nous allons le voir, il peut être exprimé sous forme d'un programme linéaire.

- ***La méthode de Frank et Wolfe : [5]***

Description de la méthode:

Cette méthode s'applique si les contraintes sont linéaires et si X est borné. Une manière simple de générer une direction d^k satisfaisant la condition de descente $d^k < 0$, si l'on pose $d^k = \bar{x}^k - x^k$, est de minimiser la dérivée directionnelle de f dans la direction d^k, comme c'est le cas pour la méthode du gradient ; mais il faut en plus prendre garde de ne pas sortir de l'ensemble des solutions admissibles (le point $\bar{x}^k$ généré doit lui-même appartenir à X, de sorte que le point $x^k + d^k$soit lui-même admissible). Le sous-problème de recherche de d^k peut être formulé ainsi :

$$\text{Minimiser} \quad \nabla f(x^k) \cdot (x - x^k)$$
$$\text{sous les contraintes} \quad x \in X$$

et nous pouvons obtenir $\bar{x}^k$ comme la solution optimale de ce sous-problème. Celui-ci est un programme linéaire (les contraintes du problème de départ le sont elles-mêmes) et peut être résolu par l'algorithme du simplexe. Selon un théorème connu,

sa solution optimale $\bar{x}^k$ se trouvera alors en l'un des points extrêmes du domaine X, de sorte que l'optimisation monodimensionnelle le long de d^k devra être faite sous la restriction $0 < a^k \leq 1$. La méthode se termine si la direction générée est égale au vecteur nul.

Convergence de la méthode:

Théorème : [5]

Considérons le problème de minimisation de $f(x)$ sous contraintes $x \in X$ où X est un polytope borné. Alors la méthode de Frank et Wolfe démarrée depuis un point initial admissible converge vers un point satisfaisant la condition de Karush-Kuhn-Tucker.

La méthode converge vers un point de Karush-Kuhn-Tucker, mais pas nécessairement de manière finie car elle peut être sujette au zigzag, à l'instar de la méthode du gradient (nous en reparlerons lors de son utilisation pratique).

- ***La méthode de Zoutendjik : [12]***

La méthode de Zoutendijk fonctionne selon le même schéma : à chaque itération, elle génère une direction de descente admissible et ensuite minimise *f* le long de cette direction. A l'image de la méthode de Frank et Wolfe, le sous-programme de recherche d'une telle direction est linéaire.

Cas des contraintes linéaires:

Les variables apparaissant dans la fonction objectif du sous-programme linéaire sont les composantes de d^k, que l'on borne afin d'empêcher le problème d'être non borné Le problème de recherche de direction finale est :

$$\begin{aligned} &\text{minimiser} \quad \nabla f(x^k) \cdot d \\ &\text{sous les contraintes} \quad \nabla h_i(x^k) \cdot d \leq 0 \quad \forall i \in I \\ &\qquad\qquad\qquad\quad -1 \leq d^i \leq 1 \quad \forall i \in \{1, \dots, n\} \end{aligned}$$

I : L'ensemble des contraintes actives en x^k: $I = \{i | h_i(x^k) = 0\}$

Cas des contraintes non linéaires:

En raison de la non linéarité de h_i, il se peut que tout déplacement dans cette direction conduise en un point non admissible. Cette approche doit donc être modifiée pour pouvoir prendre en compte ce type de contraintes.

Théorème : [12]

Considérons le problème de minimisation de $f(x)$ sous contraintes $h_i \leq 0$ pour $i = 1, \dots m$. Soient x^k une solution admissible. Supposons, de plus, que f et les h_i sont continument différentiables.

Si $\nabla f(x^k) \cdot d < 0$ et $\nabla h_i(x^k) \cdot d < 0 \ \forall i \in I$, alors d est une direction de descente admissible de f en x^k.

Pour trouver un vecteur *d* satisfaisant ces deux inégalités strictes, une possibilité est de minimiser le maximum de la valeur $\nabla f(x^k) \cdot d$, et des valeurs $\nabla h_i(x^k) \cdot d$. En dénotant ce minimum par z et en introduisant les restrictions interdisant au problème d'être non borné, nous obtenons le problème suivant :

$$\begin{aligned} &\text{minimiser} \quad z \\ &\text{sous contrainte} \quad \nabla f(x^k) \cdot d - z \leq 0 \end{aligned}$$

$$\nabla h_i(x^k) \cdot d - z \leq 0 \quad \forall i \in I$$
$$-1 \leq d^i \leq 1 \quad \forall i \in \{1, \dots, n\}$$

Si$z_{optimal} < 0$, alors *d* est manifestement une direction de descente admissible. Sinon x^k satisfait la condition de Fritz John (x^k satisfait la condition de Fritz John si et seulement si la valeur optimale de la fonction objectif du programme linéaire précédent est égale à zéro).

Convergence de la méthode: [12]

En général, la convergence de la méthode de Zoutendijk n'est pas garantie. Un contre exemple, attribué à Wolfe, a montré qu'elle pouvait ne pas converger vers un point satisfaisant les conditions de Karush-Kuhn-Tucker.

Observons qu'avec l'emploi de cette méthode, nous ne pouvons pas traiter de contraintes non linéaires qui soient sous forme d'égalités, car il ne serait pas possible de générer une direction de déplacement qui n'amène à sortir du domaine. Cette difficulté peut être surmontée en introduisant des mouvements correctifs destinés à revenir dans la région admissible.

2.2.2 Méthodes de plans sécants [5]

Les méthodes de plans sécants sont applicables si X est fermé, si les contraintes qui le définissent sont convexes (X l'est donc également) et si f est convexe. Si f est linéaire, le principe de ces méthodes est "d'enfermer" l'ensemble des solutions admissibles dans un polytope *P* formé d'au moins $n + 1$ hyperplans (c'est le nombre minimal d'hyperplans que doit comporter le polytope initial, au début de l'exécution). P définit alors un nouvel ensemble de solutions admissibles et le problème consistant à minimiser f sur P est un programme linéaire. Si sa solution optimale appartient à X, l'exécution est terminée car cette solution est également celle du problème original. Sinon, un nouvel hyperplan est introduit réduisant *P* à un nouveau polytope. Celui-ci doit répondre à deux exigences : il ne doit tout d'abord éliminer aucune solution admissible du problème original ; ensuite, la solution optimale du programme linéaire résolu à l'itération précédente doit devenir non admissible. Puis, le processus est reconduit jusqu'à satisfaire la condition d'arrêt.

- ***La méthode de Kelley : [5]***

Description de la méthode:

En plus des hypothèses formulées juste avant, nous admettons donc que f est linéaire. X est initialement enfermée dans un ensemble de contraintes linéaires qui sont habituellement choisies comme de simples bornes sur les variables. Le point obtenu à la première itération x^1 est la solution du programme linéaire correspondant. Si x^1 n'est pas une solution admissible du problème de départ, il existe au moins un i tel que $g_i(x^k) > 0$. Nous pouvons alors choisir s tel que $g_s(x^k) = max_{i \leq 1 \leq m}\ g_i(x^k) > 0$. Une nouvelle contrainte linéaire est ensuite ajoutée, donnée par l'approximation linéaire de $g_s(x^k)$ au voisinage de x^k :

$$g_s(x^k) + \nabla g_s(x^k) \cdot (x - x^k) \leq 0$$

Remarquons que ce nouvel hyperplan coupe x^k, qui ne satisfait pas la relation précédente, du domaine admissible original qui est inclus dans le demi-espace défini

par cette relation. La solution du programme linéaire précédent, auquel cette contrainte a été ajoutée, est le nouveau point x^{k+1}, puis le processus recommence jusqu'à satisfaire la condition d'arrêt.

Convergence de la méthode:

Théorème : [5]

Soit une séquence $\{x^k\}$ générée par la méthode de Kelley sous les hypothèses énoncées ci-dessus. Alors tout point limite de $\{x^k\}$ est la solution optimale globale du problème original.

2.2.3 Méthodes de pénalité [5]

- ***Méthode de pénalisation intérieure (méthode de barrière) :***

Le principe de ces méthodes réside dans la transformation d'un problème contraint en une séquence de problèmes sans contraintes, en ajoutant au coût une pénalité en cas de violation de celles-ci. Un tel sous problème est résolu à chaque itération d'une méthode de pénalité. L'appellation "pénalité intérieure" est employée car le minimum est approché depuis l'intérieur de X .Les méthodes de barrière s'appliquent aux problèmes dont l'ensemble admissible X est défini uniquement par une collection d'inégalités :

$$\begin{aligned} &\text{minimiser} \quad f(x) \\ &\text{sous les contraintes} \quad h_i(x) \leq 0, \quad i = 1, \dots, m \\ &\qquad\qquad\qquad\quad x \in \mathbb{R}^n \end{aligned}$$

La fonction de barrière, notée $B(x)$, est ajoutée à $f(x)$; elle est continue sur X_I (où $X_I = \{x | h_i(x) < 0,\ \forall i \in \{1, \dots, m\}\}$ et sa valeur tend vers l'infini lorsque la frontière de X est approchée par l'intérieur, c'est-à-dire lorsque l'un des $h_i(x)$ approche zéro par les valeurs négatives. Une itération de la méthode consiste ensuite minimiser la fonction $f(x) + \mathcal{E}B(x)$; où $\mathcal{E}$ est un paramètre réel strictement positif) à l'aide d'algorithmes de minimisation directe.
Les fonctions de barrière les plus répandues sont les suivantes :

Le logarithmique : $B(x) = -\sum_{i=1}^{m} \ln(-h_i(x))$

L'inverse : $B(x) = -\sum_{i=1}^{m} \frac{1}{h_i(x)}$

Il est important de noter que, si tous les $h_i(x)$ sont convexes, ces deux fonctions de barrière le sont également. La méthode de barrière est définie en introduisant la séquence de paramètres $\{\mathcal{E}^k\}, k = 0,1, \dots$ avec $0 < \mathcal{E}^{k+1} < E^k$ *et* $\mathcal{E}^k \to 0$ *lorsque* $k \to \infty$. Une itération de celle-ci consiste à déterminer :

$$x^k = \arg\min_{x \in X_I} \{f(x) + \mathcal{E}^k B(x)\}$$

Théorème : [5]

Tout point limite d'une séquence $\{x^k\}$ générée par une méthode de barrière est un minimum global du problème contraint original.

Remarquons qu'en dehors de x^0, les points obtenus à l'itération précédente sont, à chaque itération, utilisés comme points de départ de l'algorithme de minimisation directe (cela est valable pour les deux types de méthodes de pénalité).

- ***Méthode de pénalisation extérieure : [5]***

Les méthodes de pénalité extérieures cherchent à approcher le minimum depuis l'extérieur de X à $f(x)$, et ajouter une *fonction de pénalité extérieure* $P(x)$ dont la valeur est égale à zéro si x est admissible et supérieure à zéro s'il ne l'est pas. L'ajout de cette fonction a pour seul but de pénaliser la fonction objectif en cas de violation d'une ou de plusieurs contraintes**.** La fonction de pénalité quadratique $P(x)$ est définie comme suit :

$$P(x) = \sum_{i=1}^{m} (h_i(x))^2 u_i(h_i(x))$$

où $$u_i(h_i(x)) = \begin{cases} 0 & si\ h_i(x) \leq 0 \\ 1 & sinon \end{cases}$$

De façon analogue à la méthode de barrière, nous introduisons un paramètre μ qui permet d'amplifier ou de diminuer sa valeur et, à chaque itération, le sous-problème à résoudre à l'aide d'une méthode de minimisation directe sera de la forme suivante : minimiser $f(x) + \mu P(x)$ *avec* $x \in \mathbb{R}^n$. A chaque itération, la méthode obtient x^k de la manière suivante :

$$x^k = \arg \min_{x \in X_I} \{f(x) + \mu^k P(x)\}$$

Le comportement de la méthode de pénalité extérieure est très similaire à celui de la méthode de barrière. En particulier, des résultats théoriques équivalents peuvent être établi et assurer sa convergence. Les considérations plus pratiques, principalement lorsque $P(x)$ croît, sont également similaires à celles données pour la méthode de barrière, la seule différence notable étant que la méthode de pénalité extérieure ne nécessite aucunement un point initial intérieur à X.

II. Application aux SVMs

1. Introduction [14]

Les Support Vector Machines souvent traduit par l'appellation de Séparateur à Vaste Marge (SVM) sont une classe d'algorithmes d'apprentissage initialement définis pour la discrimination c'est-à-dire la prévision d'une variable qualitative binaire. Ils ont été ensuite généralisés à la prévision d'une variable quantitative. Dans le cas de la discrimination d'une variable dichotomique, ils ont basé sur la recherche de l'hyperplan de marge optimale qui, lorsque c'est possible, classe ou sépare correctement les données tout en étant le plus éloigné possible de toutes les observations. Le principe est donc de trouver un classifieur, ou une fonction de discrimination, dont la capacité de généralisation (qualité de prévision) est la plus grande possible. Voir [15,16,17,18].

Le principe de base des SVM consiste de ramener le problème de la discrimination à celui, linéaire, de la recherche d'un hyperplan optimal. Deux idées ou astuces permettent d'atteindre cet objectif : La première consiste à définir l'hyperplan comme solution d'un problème d'optimisation sous contraintes dont la fonction objectif ne s'exprime qu'à l'aide de produits scalaires entre vecteurs et dans lequel le nombre de contraintes "actives" ou vecteurs supports contrôle la complexité du modèle. Le passage à la recherche de surfaces séparatrices non linéaires est obtenu par l'introduction d'une fonction noyau (Kernel) dans le produit scalaire induisant implicitement une transformation non linéaire des données vers un espace intermédiaire (feature space) de plus grande dimension. D'où l'appellation couramment rencontrée de machine à noyau ou Kernel machine. Sur le plan théorique, la fonction noyau définit un espace hilbertien, dit auto-reproduisant et isométrique par la transformation non linéaire de l'espace initial et dans lequel est résolu le problème linéaire.

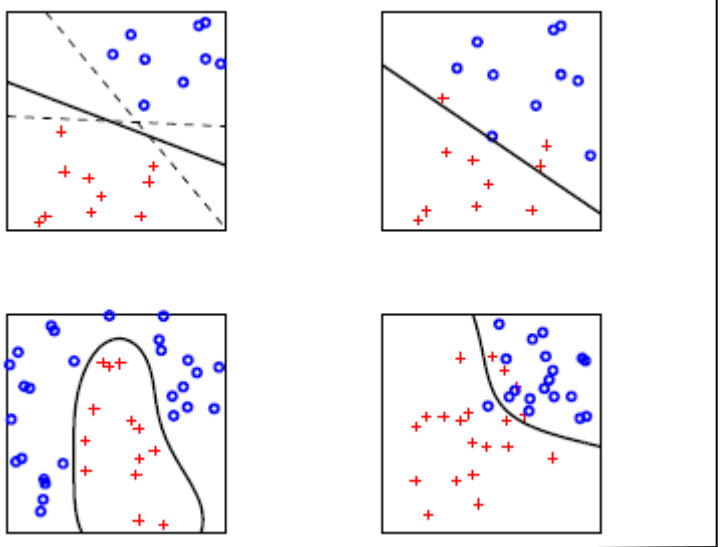


Figure 1 : Exemple de quatre types de problèmes de discrimination binaire ou il s'agit de séparer les points bleus des croix rouges. La frontière de décision est représentée en noir.

2. SVMs linéaires

2.1 La discrimination linéaire [15]

Le cas simple est le cas d'une fonction discriminante linéaire, obtenue par combinaison linéaire du vecteur d'entrée $= (x_1, \dots, x_N)^T$, avec un vecteur de poids $w = (w, \dots, w_N)^T$: $h(x) = w^T x + w_0$

Il est alors décidé que x est de classe 1 si $h(x) \geq 0$ et de classe -1 sinon. C'est un classifieur linéaire.

La frontière de décision $h(x) = 0$ est un ***hyperplan***, appelé *hyperplan séparateur*, ou *séparatrice*. Le but d'un algorithme d'apprentissage supervisé est d'apprendre la fonction $h(x)$ par le biais d'un ensemble d'apprentissage :

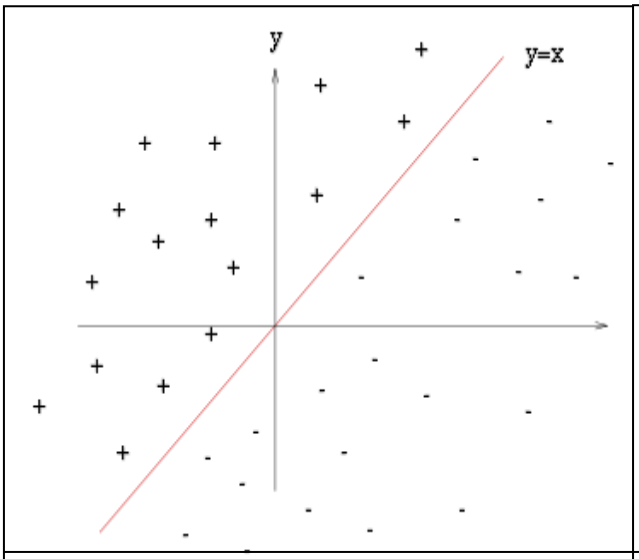
$$\{(x^1, l_1), (x^2, l_2), \dots, (x^k, l_k), \dots, (x^p, l_p)\} \subset \mathrm{R^N} \times \{-1,1\}$$

où les l_k sont les labels, p est la taille de l'ensemble d'apprentissage, N la dimension des vecteurs d'entrée. Si le problème est linéairement séparable, on doit alors avoir :

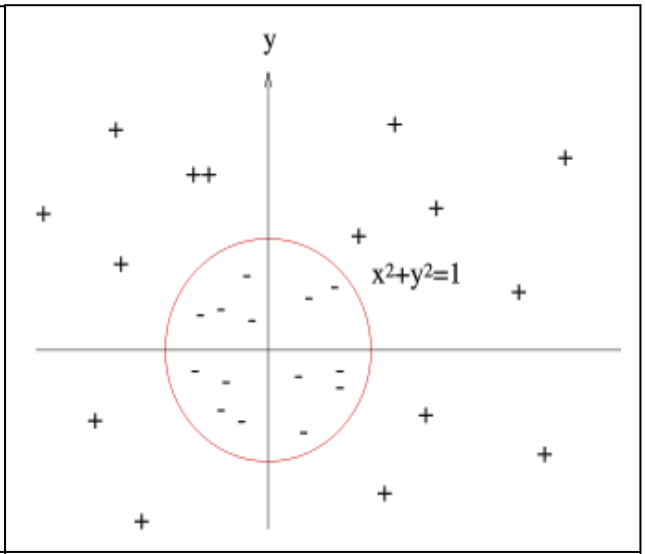
$$l_k h(x^k) \geq 0 \quad 1 \leq k \leq p \text{ autrement dit } l_k(w^T x^k + w_0)$$

Exemple: Imaginons un plan (espace à deux dimensions) dans lequel sont répartis deux groupes de points. Ces points sont associés à un groupe: les points (+) pour *y* > *x* et les points (-) pour *y* < *x*. On peut trouver un séparateur linéaire évident dans ce cas, la droite d'équation *y* = *x*. Le problème est dit *linéairement séparable*.

Pour des problèmes plus compliqués, il n'existe en général pas de séparateur linéaire. Imaginons par exemple un plan dans lequel les points (-) sont regroupés à l'intérieur d'un cercle, avec des points (+) tout autour : aucun séparateur linéaire ne peut correctement séparer les groupes : le problème n'est pas *linéairement séparable*. Il n'existe pas d'hyperplan séparateur.



Exemple d'un problème de discrimination à deux classes, avec un séparateur linéaire : la droite d'équation *y* = *x*. Le problème est *linéairement séparable*.



Exemple d'un problème de discrimination à deux classes, avec un séparateur non-linéaire : le cercle unité. Le problème *n'est pas linéairement séparable*.

2.2 La marge d'un classifieur [15]

Pour un échantillon donné, il est possible d'associer deux marges à un même classifieur linéaire : sa marge géométrique et sa marge numérique. La marge géométrique m d'un échantillon linéairement séparable est donnée par la plus petite distance d'un point de l'échantillon à la frontière de décision. La marge numérique μ est donnée elle par la plus petite valeur de la fonction de décision

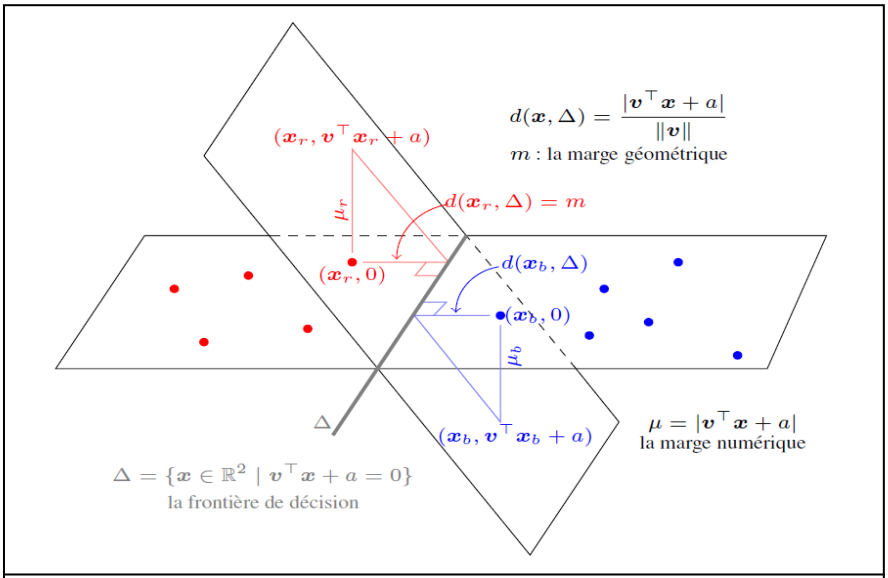


Figure 2 Illustration des deux notions de marge sur un exemple de discrimination linéaire séparable en dimension deux.

atteinte sur un point de l'échantillon. Leur définition mathématique est :

marge géométrique
$$m = \min_{i \in [1,N]} dist(x_i, \Delta(v, a))$$

marge numérique
$$\mu = \min_{i \in [1,N]} |v^T x_i + a|$$

La figure 2 illustre ces deux notions de marge pour un exemple en deux dimensions.
On voit que pour une frontière de décision donnée, la marge géométrique m est fixée alors que la marge numérique μ dépend de la « pente » de l'hyperplan de décision (donc de $\|v\|$). En effet, pour une observation donnée, les deux marges forment les cotés adjacents d'un triangle rectangle dont l'hypoténuse est définie par la fonction caractéristique $v^T x + a$.

2.3 Maximisation de la marge d'un classifieur [15]

Lorsque des observations sont linéairement séparables, comme l'illustre la figure1 (en haut à gauche) il existe dans le cas général une infinité de frontières de décision linéaires séparant cet échantillon. La notion de marge offre un critère de choix parmi toutes ces solutions en admettant que maximiser la marge c'est aussi

maximiser la confiance et donc minimiser la probabilité d'erreur associée au classifieur. Nous allons résoudre le problème suivant :

$$\max_{v,a} \underbrace{\min_{i\in[1,p]} dist(x^i, \Delta(v,a))}_{marge:m}$$

En introduisant explicitement la marge comme une variable, ce problème se réécrit comme un problème d'optimisation sous contraintes :

$$\begin{cases} \max\limits_{v,a} \ m \\ avec \ \min_{i\in[1,p]} \frac{|v^T x^i + a|}{\|v\|} \geq m \end{cases}$$

C'est un problème mal posé dans le sens ou si (v,a) est une solution, (kv, ka); $\forall\, 0 < k$ l'est aussi. Une manière de traiter cette difficulté est d'effectuer le changement de variable : $w = \frac{v}{m\|v\|}$ et $b = \frac{a}{m\|v\|}$. Le problème se réécrit alors :

$$\begin{cases} \max\limits_{v,a} \ \frac{1}{\|w\|} \\ avec \ l_i(w^T x_i + b) \geq 1 \ ; i = 1, \dots, p \end{cases}$$

Puisque $\|w\| = \frac{1}{m}$. Cela revient à fixer à un la marge numérique du classifieur recherché (celui de norme minimale). La formulation « classique » des SVM s'obtient alors en minimisant $\|w\|^2$ au lieu de maximiser l'inverse de la norme, ce qui donne le problème suivant qui admet la même solution que le problème précédent.

2.4 Formulation primale [15]

La marge est la plus petite distance entre les échantillons d'apprentissage et l'hyperplan séparateur qui satisfasse la condition de séparabilité (à savoir $l_k(w^T x^k + w_0) \geq 0$ comme expliqué précédemment). La distance d'un échantillon x^k à l'hyperplan est donnée par sa projection orthogonale sur le vecteur de poids.

$$\frac{l_k(w^T x^k + w_0)}{\|w\|}$$

L'hyperplan séparateur (w,w_0) de marge maximale est donc donné par :

$$arg\ \max_{w,w_0} \left\{ \frac{1}{\|w\|} \min_k [l_k(w^T x^k + w_0)] \right\}$$

Afin de faciliter l'optimisation, on choisit de normaliser w et w_0, de telle manière que les échantillons à la marge (x^+_{marge} pour les vecteurs supports sur la frontière positive, et x^-_{marge} pour ceux situés sur la frontière opposée) satisfassent :

$$\begin{cases} w^T x^+_{marge} + \mathrm{w}_0 = 1 \\ w^T x^-_{marge} + \mathrm{w}_0 = -1 \end{cases}$$

D'où pour tous les échantillons, $k = 1, \dots, p$

$$l_k(w^T x^k + w_0) \geq 1$$

Cette normalisation est parfois appelée la forme canonique de l'hyperplan, ou *hyperplan canonique*.

Avec cette mise à l'échelle, la marge vaut désormais $\frac{1}{\|w\|}$, il s'agit donc de maximiser $\|w\|^{-1}$. La formulation dite *primale* des SVM s'exprime alors sous la forme suivante:

$$\text{Minimiser } \frac{1}{2}\|w\|^2 \text{ sous les contraintes } l_k(w^T x^k + w_0) \geq 1$$

Ceci peut se résoudre par la méthode classique des multiplicateurs de Lagrange, où le lagrangien est donné par :

$$L(w, w_0, \mathrm{a}) = \frac{1}{2}\|w\|^2 - \sum_{\mathrm{k}=1}^{\mathrm{p}} \mathrm{a_k}\{l_k(w^T x^k + w_0) - 1\} \qquad (1)$$

Le lagrangien doit être minimisé par rapport à w et w_0, et maximisé par rapport à a.

2.5 Formulation duale [14]

En annulant les dérivées partielles du lagrangien, selon les conditions de Karush-Karush-Kuhn-Tucker, on obtient :

$$\begin{cases} \sum_{\mathrm{k}=1}^{\mathrm{p}} a_{\mathrm{k}} \mathrm{l_k} x^k = \mathrm{w} \\ \sum_{\mathrm{k}=1}^{\mathrm{p}} a_{\mathrm{k}} \mathrm{l_k} = 0 \end{cases}$$

En réinjectant ces valeurs dans l'équation (1), on obtient la *formulation duale* :

$$\text{Maximiser } \tilde{L}(a) = \sum_{\mathrm{k}=1}^{\mathrm{p}} a_{\mathrm{k}} - \frac{1}{2}\sum_{\mathrm{i,j}} a_{\mathrm{i}} a_{\mathrm{j}} \mathrm{l_i l_j x_i^T x_j} \qquad (2)$$

$$\text{Sous les contraintes } a_k \geq 0, \text{ et } \sum_{k=1}^{p} a_k l_k = 0$$

Ce qui donne les multiplicateurs de Lagrange optimaux a_k^*.

Afin d'obtenir l'hyperplan solution, on remplace w par sa valeur optimale w^*, dans l'équation de l'hyperplan $h(x)$, ce qui donne :

$$h(x) = \sum_{k=1}^{p} a_k^* l_k (x \cdot x^k) + w_0$$

3. SVMs non linéaires – Les noyaux

3.1 Définition des noyaux [14]

L'utilisation de noyaux permet une autre généralisation très utile dans la pratique : la définition des SVM sur des objets complexes comme des images, des graphes, des protéines où des automates pour ne citer qu'eux. Nous n'avons donc pas besoin de faire d'hypothèse sur la nature du domaine des observations X et un noyau k est défini de manière générale comme une fonction de deux variables sur $\mathbb{R}$:

$$k: X, X \rightarrow \mathbb{R} \qquad (x, x') \rightarrow k(x, x')$$

Définition 4.1 La matrice de Gram du noyau $(k.,.)$ pour les observations $\{x^1, \dots, x^i, \dots, x^p\}$(pour tout entier p fini) est la matrice carrée K de taille n et de terme général $K_{i,j} = k(x, x')$

Définition 4.2 Un noyau k est dit positif si, pour tout entier n fini et pour toutes les suites de n observations possibles $(x_i,\ i = 1, \dots, n)$ la matrice de Gram associée est une matrice symétrique définie positive. L'intérêt des noyaux positifs c'est qu'il est possible de leur associer un produit scalaire. Les noyaux dits de Mercer sont des noyaux positifs définis sur un ensemble compact.

3.2 Principe [14]

La notion de marge maximale et la procédure de recherche de l'hyperplan séparateur telles que présentées pour l'instant ne permettent de résoudre que des problèmes de discrimination linéairement séparables. C'est une limitation sévère qui condamne à ne pouvoir résoudre que des problèmes particuliers. Afin de remédier au problème de l'absence de séparateur linéaire, l'idée de l'astuce du noyau (en anglais *Kernel trick*) est de reconsidérer le problème dans un espace de dimension supérieure, éventuellement de dimension infinie. Dans ce nouvel espace, il est alors probable qu'il existe une séparation linéaire.

Plus formellement, on applique aux vecteurs d'entrée x une transformation non-linéaire.

L'espace d'arrivée $\emptyset(X)$ est appelé *espace de redescription*. Dans cet espace, on cherche alors l'hyperplan :

$$h(x) = w^T \emptyset(x) + w_0$$

Qui vérifie

$l_k h(x^k) > 0$, pour tous les points x^k de l'ensemble d'apprentissage, c'est-à-dire l'hyperplan séparateur dans l'espace de redescription.

En utilisant la même procédure que dans le cas sans transformation, on aboutit au problème d'optimisation suivant :

Maximiser $\tilde{L}(a) = \sum_{k=1}^{p} a_k - \frac{1}{2}\sum_{i,j} a_i a_j l_i l_j \emptyset(x_i)^T \emptyset(x_j)$ (2)

Sous les contraintes $a_i \geq 0$, et $\sum_{k=1}^{p} a_k l_k = 0$

Le problème de cette formulation est qu'elle implique un produit scalaire entre vecteurs dans l'espace de redescription, de dimension élevée, ce qui est couteux en termes de calculs. Pour résoudre ce problème, on utilise une astuce connue sous le nom de Kernel trick, qui consiste à utiliser une fonction noyau, qui vérifie :

$K(x_i, x_j) = \emptyset(x_i)^T \cdot \emptyset(x_j)$

d'où l'expression de l'hyperplan séparateur en fonction de la fonction noyau :

$$h(x) = \sum_{k=1}^{p} a_k^* l_k K(x^k, x) + w_0 = 0$$

L'intérêt de la fonction noyau est double :

- Le calcul se fait dans l'espace d'origine, ceci est beaucoup moins coûteux qu'un produit scalaire en grande dimension.
- La transformation ∅ n'a pas besoin d'être connue explicitement, seule la fonction noyau intervient dans les calculs. On peut donc envisager des transformations complexes, et même des espaces de redescription de dimension infinie.

3.3 Choix de la fonction noyau

En pratique, on ne connaît pas la transformation ∅, on construit plutôt directement une fonction noyau. Celle-ci doit respecter certaines conditions, elle doit correspondre à un produit scalaire dans un espace de grande dimension. Le théorème de Mercer explicite les conditions que K doit satisfaire pour être une fonction noyau : elle doit être symétrique, semi-définie positive.

L'exemple le plus simple de fonction noyau est le noyau linéaire :

$$K(x^i, x^j) = {x^i}^{\mathrm{T}} \cdot x^j$$

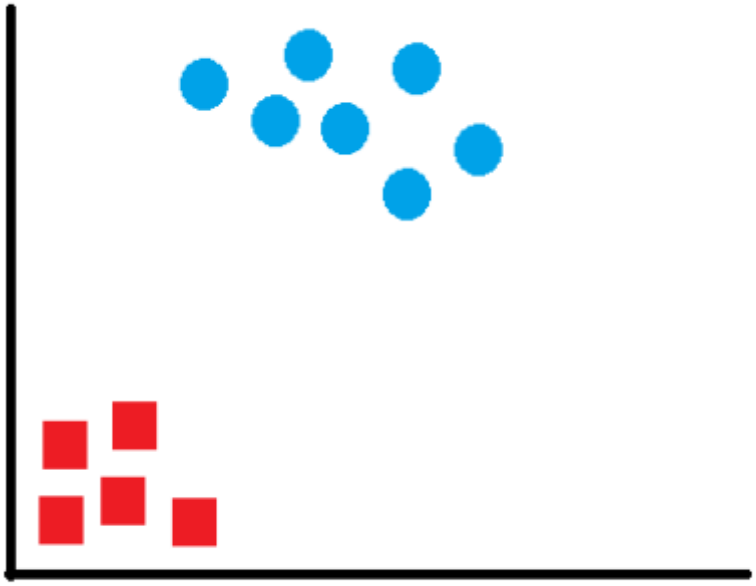
On se ramène donc au cas d'un classifieur linéaire, sans changement d'espace. L'approche par Kernel trick généralise ainsi l'approche linéaire. Le noyau linéaire est parfois employé pour évaluer la difficulté d'un problème.

Des noyaux usuels employés avec les SVM sont :

- le noyau polynomial $K(x^i, x^j) = ({x^i}^{\mathrm{T}} \cdot x^j + 1)^d$
- le noyau gaussien $K(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$

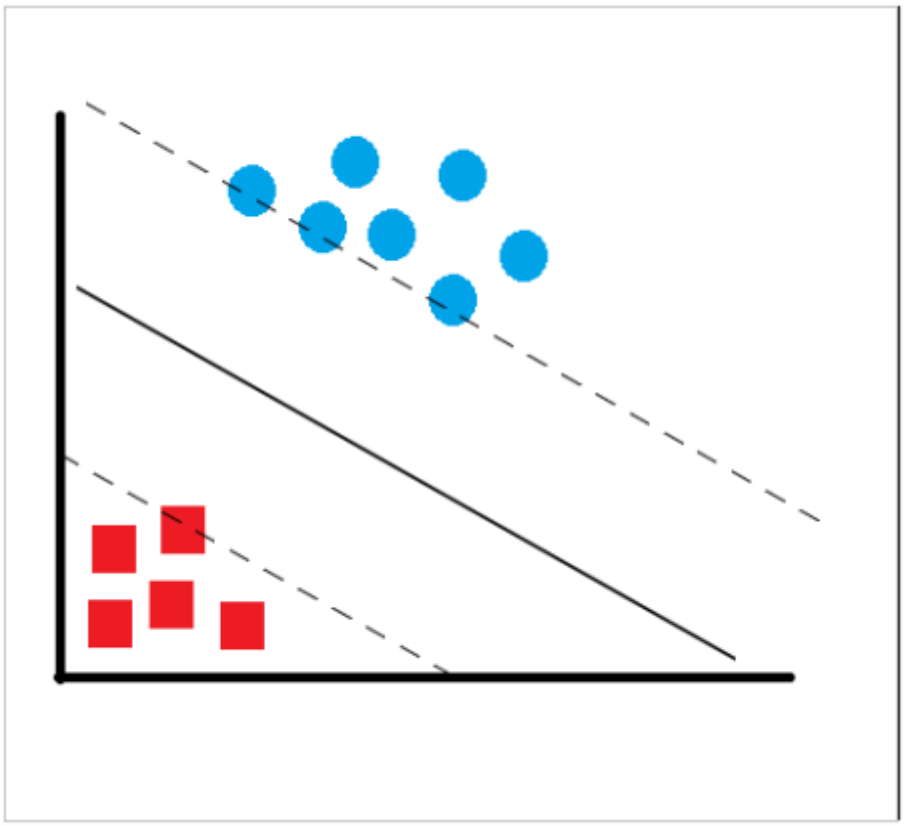
III. Implémentation

Support Vector Machines (SVM) est une méthode de classification des données qui sépare les données à l'aide d'hyperplans. Le concept de SVM est très intuitif et facilement compréhensible. Si nous avons des données étiquetées, SVM peut être utilisé pour générer plusieurs hyperplans de séparation de sorte que l'espace de données soit divisé en segments et que chaque segment ne contienne qu'un seul type de données. La technique SVM est généralement utile pour les données qui ont une non-régularité, c'est-à-dire des données dont la distribution est inconnue.
Prenons un exemple simple pour comprendre comment fonctionne SVM. Disons que vous n'avez que deux types de valeurs et que nous pouvons les représenter comme dans la figure:



Ces données sont simples à classer et on peut voir que les données sont clairement séparées en deux segments. Toute ligne qui sépare les éléments rouges et bleus peut être utilisée pour classer les données ci-dessus. Si ces données étaient des données multidimensionnelles, tout plan peut séparer et classer avec succès les données. Cependant, nous voulons trouver la solution «la plus optimale». Quelle sera alors la caractéristique de cette ligne la plus optimale? Nous devons nous rappeler que ce ne sont que des données de formation et que nous pouvons avoir plus de points de données qui peuvent se trouver n'importe où dans le sous-espace. Si notre ligne est trop proche de l'un des points de données, les données de test bruyantes sont plus susceptibles d'être classées dans un segment incorrect. Nous devons choisir la ligne qui se situe entre ces groupes et est la plus éloignée de chacun des segments.

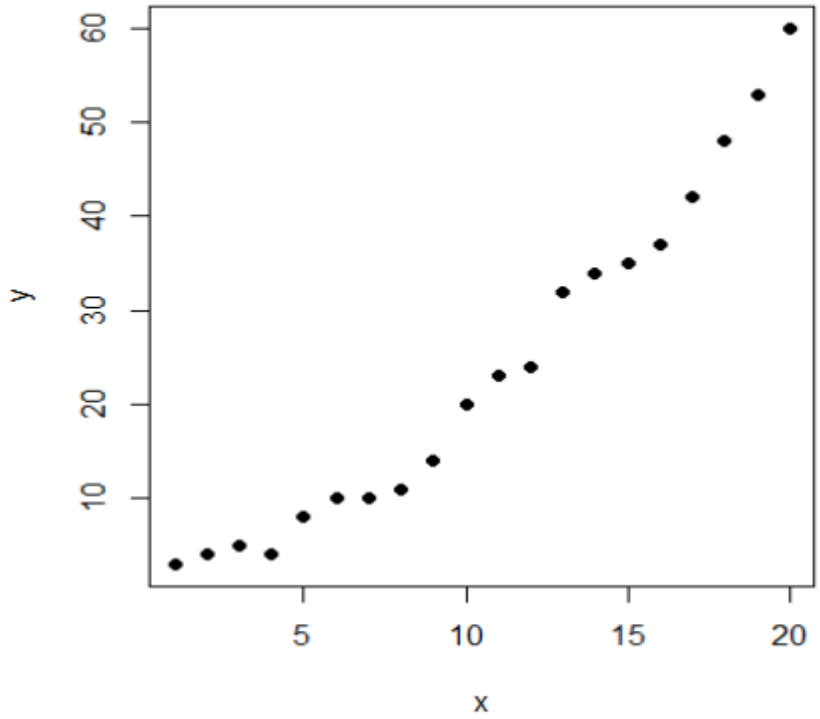
Pour résoudre cette ligne de classificateur, nous dessinons la ligne comme y = ax + b et la rendons équidistante des points de données respectifs les plus proches de la ligne. Nous voulons donc maximiser la marge (m).



Nous savons que tous les points sur la ligne ax + b = 0 satisferont l'équation. Si on trace deux lignes parallèles - ax + b = -1 pour un segment et ax + b = 1 pour l'autre segment tel que ces lignes passent par un point de donnée dans le segment le plus proche de notre ligne alors la distance entre ces deux lignes sera notre marge. Par conséquent, notre marge sera m = 2 / || a ||. En d'autres termes, si nous avons un ensemble de données d'apprentissage (x1, x2, x3… xn) et que nous voulons produire et aboutir y tel que y soit -1 ou 1 (selon le segment auquel appartient le point de données), alors notre classificateur devrait classer correctement les points de données sous la forme de y = ax + b. Cela signifie également que y (ax + b)> 1 pour tous les points de données. Comme il faut maximiser la marge, on résout ce problème avec la contrainte de maximiser 2 / || a || ou, minimisant || a || ^ 2/2 qui est fondamentalement la forme double de la contrainte. La résolution de ce problème peut être facile ou complexe en fonction de la dimension des données. Cependant, nous pouvons le faire rapidement avec R et essayer quelques exemples de données.

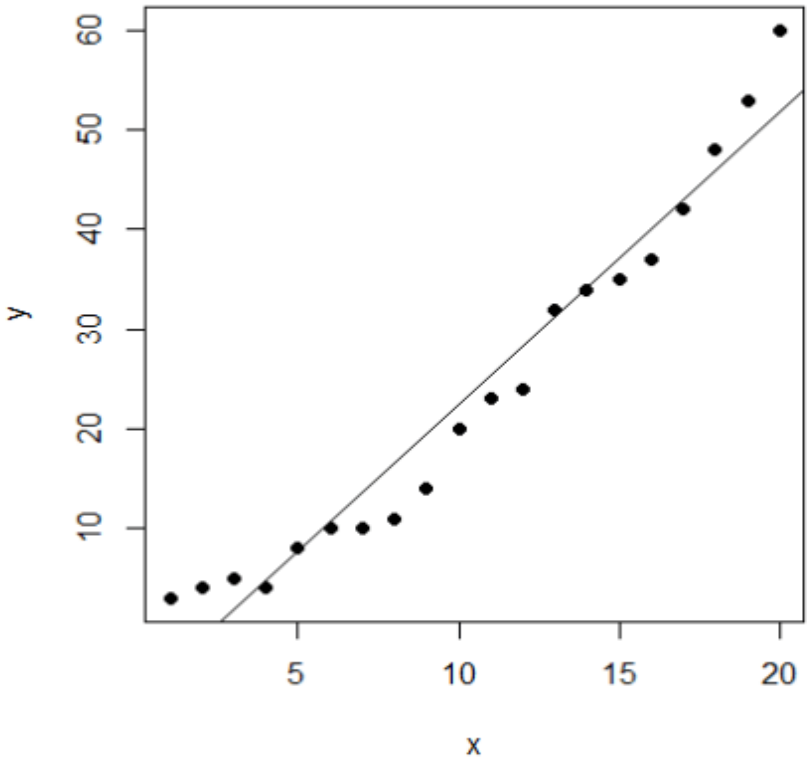
Construire un SVM en utilisant le langage R, voir [19]

```
> x=c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)
> y=c(3,4,5,4,8,10,10,11,14,20,23,24,32,34,35,37,42,48,53,60)
> # Créer un bloc de données des données
> train=data.frame(x,y)
> # Tracer le jeu de données
> plot(train,pch=16)
```



Semble être une assez bonne donnée. En regardant la figure ci-dessus, il semble également qu'une régression linéaire devrait également convenir aux données. Nous utiliserons les deux modèles et les comparerons. Tout d'abord, le code pour la régression linéaire:

```
>  # Régression linéaire
>  model <- lm(y ~ x, train)
>  print(model)

        Call:
        lm(formula = y ~ x, data = train)

        Coefficients:
        (Intercept)          x
                -6.979       2.936

>  # Tracer le modèle en utilisant abline
>  abline(model)
```



SVM

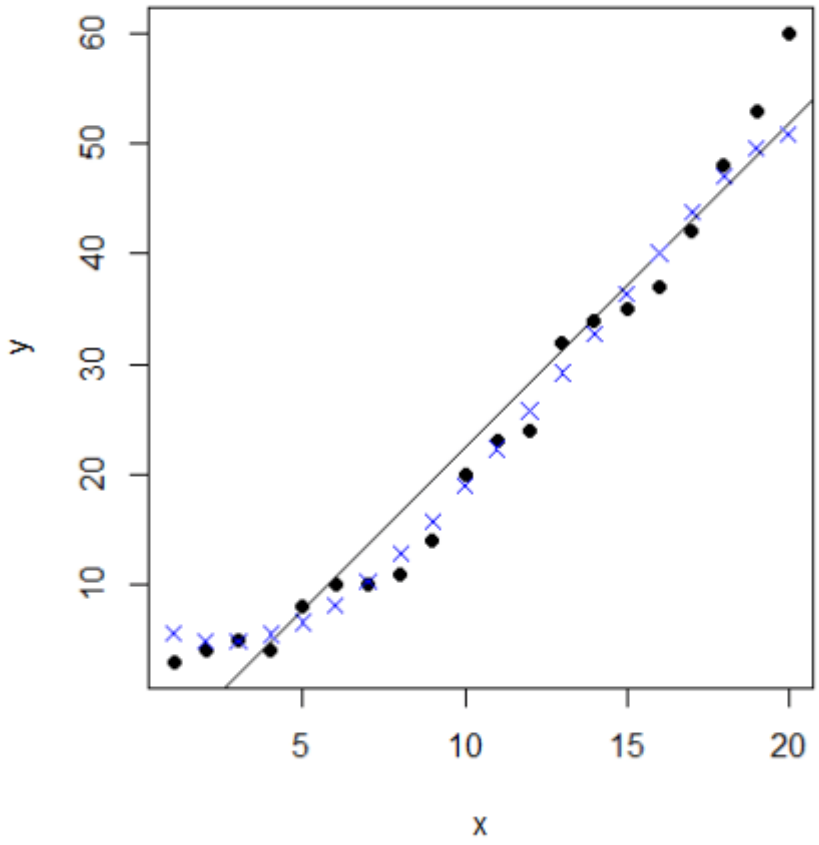
Pour utiliser SVM en R, nous avons un package e1071. Le paquet n'est pas préinstallé, il faut donc exécuter la ligne «install.packages (" e1071 ", dep = TRUE) pour installer le paquet, puis importer le contenu du paquet en utilisant la commande library. La syntaxe de SVM package est assez similaire à la régression linéaire. Nous utilisons la fonction SVM ici.

```
>  #SVM
>  library(e1071)
>  # Adapter un modèle. La syntaxe de la fonction est très similaire à la fonction lm
>  model_svm <- svm(y ~ x , train)
>  print(model_svm)

Call:
svm(formula = y ~ x, data = train)

Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  1
    epsilon:  0.1

Number of Support Vectors:  9
```

```
> # Utiliser les prédictions sur les données
> pred <- predict(model_svm, train)
> print(pred)
```

```
       1        2        3        4        5        6        7
5.571694 4.916612 4.931428 5.533846 6.645360 8.226929 10.275791
       8        9       10       11       12       13       14
12.788151 15.715255 18.946717 22.338608 25.773435 29.213435 32.705881
      15       16       17       18       19       20
36.324029 40.067146 43.773434 47.100688 49.597937 50.844404
```

```
> # Tracez les prédictions et l'intrigue pour voir notre ajustement de modèle
> points(train$x, pred, col = "blue", pch=4)
```



Les points suivent les valeurs réelles beaucoup plus étroitement que l'abline. Peut-on vérifier cela? Calculons les erreurs quadratiques moyennes (RMSE) pour les deux modèles:

```
>   # Le modèle linéaire a une partie résiduelle que nous pouvons extraire et calculer directement RMSE
>  error <- model$residuals
>  print(error)

     1          2          3          4          5          6          7
7.0428571  5.1067669  3.1706767 -0.7654135  0.2984962 -0.6375940 -3.5736842
     8          9         10         11         12         13         14
-5.5097744 -5.4458647 -2.3819549 -2.3180451 -4.2541353  0.8097744 -0.1263158
    15         16         17         18         19         20
-2.0624060 -2.9984962 -0.9345865  2.1293233  4.1932331  8.2571429

>  lm_error <- sqrt(mean(error^2))
>  print(lm_error)

[1] 3.832974

>   # Pour svm, nous devons calculer manuellement la différence entre les valeurs réelles (train $ y) avec nos prédictions (pred)
>  error_2 <- train$y - pred
>  print(error_2)

         1          2          3          4          5          6
-2.57169448 -0.91661221  0.06857208 -1.53384558  1.35464022  1.77307142
         7          8          9         10         11         12
-0.27579137 -1.78815067 -1.71525549  1.05328265  0.66139165 -1.77343473
        13         14         15         16         17         18
 2.78656450  1.29411917 -1.32402940 -3.06714591 -1.77343435  0.89931169
        19         20
3.40206268  9.15559591

>  svm_error <- sqrt(mean(error_2^2))
>  print(svm_error)

[1] 2.698615
```

Dans ce cas, le RMSE pour le modèle linéaire est ~ 3,83 alors que notre modèle SVM a une erreur inférieure à ~ 2,7. Une implémentation simple de SVM a une précision supérieure à celle du modèle de régression linéaire. Cependant, le modèle SVM va bien au-delà. Nous pouvons encore améliorer notre modèle SVM et le régler de manière à ce que l'erreur soit encore plus faible. Nous allons maintenant approfondir la fonction SVM et la fonction de réglage. Nous pouvons spécifier les valeurs pour le paramètre de coût et epsilon qui est 0.1 par défaut. Un moyen simple est d'essayer pour chaque valeur d'epsilon entre 0 et 1 (je prendrai des mesures de 0,01) et essaye de même pour la fonction de coût de 4 à 2 ^ 9 (je prendrai des pas exponentiels de 2 ici). Je prends 101 valeurs d'epsilon et 8 valeurs de fonction de coût. Je vais donc tester 808 modèles et voir ceux qui fonctionnent le mieux. Le code peut prendre quelques instants pour exécuter tous les modèles et trouver la meilleure version. Le code correspondant sera

```
>  svm_tune <- tune(svm, y ~ x, data = train, ranges = list(epsilon = seq(0,1,0.01), cost = 2^(2:9)))
>  print(svm_tune)
```

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 epsilon cost
 0.02 16

- best performance: 2.608173

```
>  # Cette meilleure performance dénote le MSE. Le RMSE correspondant est 1.614985 qui est la racine carrée de MSE
```

Un avantage de l'accordage dans R est qu'il nous permet d'extraire directement la meilleure fonction. Nous n'avons rien à faire et il suffit d’extraire la meilleure fonction de la liste svm_tune. Nous pouvons maintenant voir l'amélioration de notre modèle en calculant son erreur RMSE en utilisant le code suivant

```
>  # Le meilleur modèle
>  best_mod <- svm_tune$best.model
>  print(best_mod)
```

```
Call:
best.tune(method = svm, train.x = y ~ x, data = train, ranges = list(epsilon = seq(0,
    1, 0.01), cost = 2^(2:9)))

Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  16
      gamma:  1
    epsilon:  0.02

Number of Support Vectors:  19
```

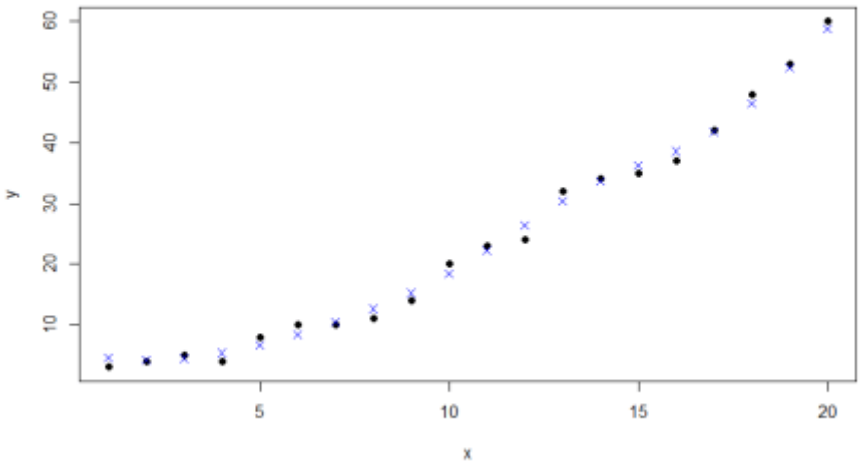
```
>  best_mod_pred <- predict(best_mod, train)
>  print(best_mod_pred)

     1          2          3          4          5          6          7
3.353137   3.635265   4.835531   6.344054   7.646535   8.623836   9.648662
     8          9         10         11         12         13         14
11.380649 -14.345725  18.546186  23.363298  27.850943  31.270251  33.545553
    15         16         17         18         19         20
35.352123  37.763427  41.650400  47.168230  53.604463  59.648663

>  error_best_mod <- train$y - best_mod_pred
>  print(error_best_mod)
```

```
       1          2          3          4          5          6          7 
-0.3531372  0.3647349  0.1644691 -2.3440536  0.3534648  1.3761640  0.3513385 
       8          9         10         11         12         13         14 
-0.3806491 -0.3457249  1.4538139 -0.3632976 -3.8509431  0.7297489  0.4544466 
      15         16         17         18         19         20 
-0.3521233 -0.7634274  0.3495999  0.8317696 -0.6044628  0.3513374 
```

```
> # cette valeur peut être différente sur votre ordinateur,  parce que la méthode de réglage mélange aléatoirement les données
> best_mod_RMSE <- sqrt(mean(error_best_mod^2))
> print(best_mod_RMSE)
[1] 1.183429
```

Cette méthode de réglage est appelée recherche de grille. R exécute tous les différents modèles avec toutes les valeurs possibles de la fonction epsilon et de la fonction de coût dans la plage spécifiée et nous donne le modèle qui présente l'erreur la plus faible. Voyons à quoi ressemble le meilleur modèle une fois tracé.

```
> plot(train,pch=16)
> points(train$x, best_mod_pred, col = "blue", pch=4)
```



Visuellement, les points prédits par notre modèle optimisé suivent presque les données. C'est la puissance de SVM et nous ne voyons que cela pour les données avec deux fonctionnalités. Imaginez les capacités du modèle avec plus de fonctionnalités complexes!

Conclusion

SVM est une technique puissante et particulièrement utile pour les données dont la distribution est inconnue (également appelée non-régularité dans les données). L'exemple considéré ici ne comportant que deux caractéristiques, le SVM ajusté par R est également appelé SVM linéaire. SVM est alimenté par un noyau pour gérer différents types de données et son noyau peut également être défini lors du réglage du modèle. Parmi ces exemples, citons le gaussien et le radial. Par conséquent, SVM peut également être utilisé pour des données non linéaires et ne nécessite aucune hypothèse sur sa forme fonctionnelle. Étant donné que nous séparons les données avec la marge maximale possible, le modèle devient très robuste et peut gérer des incohérences telles que des données de test bruyantes ou des données de train biaisées. Nous pouvons également interpréter les résultats produits par SVM via la visualisation. Un inconvénient commun avec SVM est associé à son réglage. Le niveau de précision dans la prévision des données de formation doit être défini dans nos données. Étant donné que notre exemple était des données générées sur mesure, nous avons tenté de rendre notre modèle aussi précis que possible en réduisant les erreurs. Cependant, dans les situations commerciales où il est nécessaire de former le modèle et de prévoir en permanence les données de test, le SVM peut tomber dans le piège du sur-ajustement. C'est la raison pour laquelle SVM doit être soigneusement modélisé, sinon la précision du modèle pourrait ne pas être satisfaisante. Comme je l'ai fait dans l'exemple, la technique SVM est étroitement liée à la technique de régression. Pour les données linéaires, nous pouvons comparer la SVM à la régression linéaire alors que la SVM non linéaire est comparable à la régression logistique. À mesure que les données deviennent de plus en plus linéaires, la régression linéaire devient de plus en plus précise. Dans le même temps, SVM réglé peut également adapter les données. Cependant, le bruit et les préjugés peuvent avoir de graves répercussions sur la capacité de régression. Dans de tels cas, SVM est très utile!

Le code R complet utilisé dans le travail est présenté sous:

```
1   x=c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)
2   y=c(3,4,5,4,8,10,10,11,14,20,23,24,32,34,35,37,42,48,53,60)
3   # Créer un bloc de données des données
4   train=data.frame(x,y)
5   # Tracer le jeu de données
6   plot(train,pch=16)
7   # Régression linéaire
8   model <- lm(y ~ x, train)
9   # Tracer le modèle en utilisant abline
10  abline(model)
11  #SVM
12  library(e1071)
13  # Adapter un modèle. La syntaxe de la fonction est très similaire à la fonction lm
14  model_svm <- svm(y ~ x , train)
15  # Utiliser les prédictions sur les données
16  pred <- predict(model_svm, train)
17  # Tracez les prédictions et l'intrigue pour voir notre ajustement de modèle
18  points(train$x, pred, col = "blue", pch=4)
```

```
# Le modèle linéaire a une partie résiduelle que nous pouvons extraire et calculer directement RMSE
error <- model$residuals
lm_error <- sqrt(mean(error^2))  # 3.832974
# Pour svm, nous devons calculer manuellement la différence entre les valeurs réelles (train $ y) avec nos prédictions (pred)
error_2 <- train$y - pred
svm_error <- sqrt(mean(error_2^2))  # 2.698615
svm_tune <- tune(svm, y ~ x, data = train, ranges = list(epsilon = seq(0,1,0.01),
 cost = 2^(2:9)))
# Cette meilleure performance dénote le MSE. Le RMSE correspondant
est 1.602951 qui est la racine carrée de MSE
# Le meilleur modèle
best_mod <- svm_tune$best.model
best_mod_pred <- predict(best_mod, train)
error_best_mod <- train$y - best_mod_pred
# cette valeur peut être différente sur votre ordinateur,  parce que la méthode de réglage mélange aléatoirement les données
best_mod_RMSE <- sqrt(mean(error_best_mod^2)) # 1.183429
plot(train,pch=16)
points(train$x, best_mod_pred, col = "blue", pch=4)
```

Conclusion générale

Cette étude nous a permis de constater que la méthode des SVM est aisée d'emploi, les logiciels de cette dernière sont disponibles sur Internet et que les expériences sont faciles à réaliser. On obtient souvent, en les utilisant, des résultats comparables à ceux obtenus avec d'autres techniques et les paramètres à régler sont moins nombreux. Cette méthode est applicable pour des tâches de classification à deux classes.

Par ailleurs, du point de vue conceptuel, la notion de marge a renouvelé la vision des méthodes inductives en général, et a stimulé tout un courant de recherche dont on peut espérer qu'il débouchera sur de nouvelles classes de méthodes, encore mieux comprises et encore plus adaptables à nos problèmes.

En perspective, on peut généraliser ce travail à la classification multiclasse.

Bibliographie

[1] J.Bass. Cours de Mathématiques, t2. Masson, (1968).

[2] AIDENE, M, OUKACHA, B, Recherche opérationnelle, Programmation linéaire,UMMTO, 2005.

[3] J. Amaya, J.A. Géomez, Strong duality for inexact linear programming problems, Optimisation 49(2001)243 ¡ 269.

[4] J.C. Culioli, Introduction à l'optimisation, Ellipses, Paris :1994

[5] M.Minoux. Programmation mathématique. Dunod, (1983).

[6] Avriel, Mordecai (2003), *Nonlinear Programming: Analysis and Methods*. Dover Publishing.(ISBN 0-486-43227-0).

[7] Xavier Antoine, Pierre Dreyfuss et Yannick Privat, ENSMN-ENSEM 2A Introduction `a l'Optimisation : aspects Théoriques, Numériques et Algorithmes. (2006-2007).

[8] Bazaraa, Mokhtar et Shetty (1979), *Nonlinear programming. Theory and algorithms*. John Wiley & Sons.(ISBN 0-471-78610-1).

[9] K . Karmanov. Programmation mathématique, édition Mir.Moscou 1977.

[10] R. Fletcher, Practical Methods of Optimization, Second Edition, John Wiley, Chichester and New York : 1987.

[11] Bonnans, J. F et Shapiro, A. (2000), *Perturbation analysis of optimization problems*. Springer.(ISBN 978-0-387-98705-7).

[12] G. Zoutendijk, *Mathematical Programming Methods*, North Holland, 1976.

[13] Nocedal, Jorge et Wright, Stephen (1999), *Numerical Optimization*. Springer. (ISBN 0-387-98793-2).

[14] John Shawe-Taylor, Nello Cristianini, *Support Vector Machines and other kernel-based learning methods*, Cambridge University Press, 2000.

[15] Martin Law "A simple introduction to support vector machines, lecteure foe CSE 802. Departement of computer Science and Engineering. Michigan State University 2011.

[16] Christopher M. Bishop, *Pattern Recognition And Machine Learning*, Springer, 2006 (ISBN 0-387-31073-8).

[17] Jean Beney, *Classification supervisée de documents : théorie et pratique*, Hermes Science, février 2008, 184p.

[18] Antoine Cornuéjols, Laurent Miclet, Yves Kodratoff, *Apprentissage Artificiel : Concepts et algorithmes*, Eyrolles, 2002 (ISBN 2-212-11020-0)

[19] Husson F, Pagès J et Lê S, Analyse de données avec R, Presses Universitaires de Rennes

Résumé :

L’optimisation est un outil important en sciences appliquées et pour l’analyse des systèmes physiques. Elle cherche `a améliorer une performance en se rapprochant d’un point optimal une fois qu’on a bien identifier l’objectif qui peut être le profit, le temps, l’énergie potentielle ou n’importe quelle quantité ou combinaison de qualité. Notre but est de trouver les valeurs des variables qui optimisent l’objectif.

Support Vecteur Machines (SVM) est une méthode de classification des données qui les sépare à l’aide d’hyperplans. Cette méthode est une technique puissante et particulièrement utile pour les données dont la distribution est inconnue. Les SVMs reposent sur deux idées clés : la notion de marge maximale et la notion de fonction noyau. Ces deux notions existaient depuis plusieurs années avant qu'elles ne soient mises en commun pour construire les SVMs.

Mots clés :

Optimisation non linéaire, optimisation sans contraintes, optimisation sous contraintes, conditions d’optimalité, méthodes d’optimisation, Support Vecteur Machines (SVM), Noyaux.