République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abderrahmane Mira de Béjaïa

Faculté des Sciences Exactes
Département de Recherche Opérationnelle

Mémoire de Master en Mathématiques Appliquées
Option : SCIENCES DE DONNES ET AIDE A LA DECISION

Thème

Apprentissage des SVMs régularisés par la méthode directe du support

Présenté par :

Mlle. SAHLI AHLEM & Mlle. REDOUANE SOUMIA

Encadré par :

Dr.BRAHMI BELKACEM

Soutenu devant le jury composé de :

Président	Dr. SOUFIT MASSINISSA	M.C.A	U. A/Mira Béjaïa
Rapporteur	Dr. BRAHMI BELKACEM	M.C.A	U. A/Mira Béjaïa
Examinateur	Dr. TOUATI SOUFIANE	M.C.A	U. A/Mira Béjaïa
Examinatrice	Mme. KENDI SALIMA	M.A.A	U. A/Mira Béjaïa

2023 – 2024

Remerciments

Tout d'abord et avant tout, nous remercions **ALLAH**, le tout-puissant, pour la volonté, la santé et la patience qu'il nous a données durant ces longues années d'études, ainsi que pour le courage qui nous a permis de terminer ce mémoire.

Nous exprimons notre sincère gratitude envers notre promoteur **Mr Brahmi Belkacem** pour son soutien constant, ses conseils avisés et son encadrement attentif tout au long de cette période.

Nous tenons également à remercier chaleureusement les **professeurs**, les **enseignants** et le personnel du département de **Recherche opérationnelle**, ainsi que les étudiants avec lesquels nous avons eu le plaisir de collaborer durant notre parcours universitaire.

Nos remerciements vont également à tous les membres du **jury** pour le temps précieux qu'ils ont consacré à l'évaluation de notre mémoire.

Enfin, nous tenons à exprimer notre profonde reconnaissance envers toutes les **personnes** qui ont contribué de près ou de loin à la réalisation de ce projet.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations :

PQC : Programmation Quadratique Convexe.
PQ : Problème Quadratique.
QC : Condition de qualification des contraintes.
SVM : Support vecteur Machine.
CSVM : SVM à Marge Souple.
KKT : Karush-Kuhn-Tucker.
SR : Solution réalisable.
SRS : Solution réalisable de Support.
RBF : Radial Basis Function.
SMO : Algorithme d'optimisation minimale séquentielle.
MDS : Méthode Direct de Support.

Introduction générale

Ce mémoire porte sur l'application, puis l'implémentation sur PYTHON de la méthode directe de support pour la résolution des problèmes de classification par des machines à vecteurs de support (SVM). Les SVMs sont des outils puissants de machine learning, développés pour résoudre des problèmes de classification, puis étendus aux tâches de régression et de clustering. Les SVMs ont montré leur efficacité dans la pratique et reposent sur des principes d'optimisation mathématique.

La programmation mathématique est une branche de l'optimisation qui s'occupe de la minimisation ou de la maximisation sous (ou sans) contraintes d'une fonction à plusieurs variables, ainsi que de la conception et de la mise en œuvre des algorithmes de résolution. La résolution numérique d'un problème quadratique nécessite l'utilisation de méthodes spécifiques. La plupart de ces méthodes sont itératives et génèrent une suite de solutions. Les itérations successives sont construites de façon à faire converger cette suite vers la solution optimale du problème.

Vapnik et ses collègues [9] ont introduit en 1995 les Machines à Vecteurs de Support, également appelées Séparateurs à Vaste Marge ou plus communément Support Vector Machines en anglais. Les SVMs sont des méthodes d'apprentissage supervisé utilisées principalement pour les problèmes de classification binaire. La construction d'un classifieur SVM binaire consiste à trouver un hyperplan optimal qui sépare les données des deux classes avec la plus grande marge possible. Les SVMs peuvent être étendues à des problèmes non linéaires grâce à l'utilisation de fonctions noyaux qui permettent de projeter les données d'entrées dans un espace de dimension supérieure (voire infinie), où un hyperplan linéaire peut être trouvé.

Les SVMs sont devenues une technique d'apprentissage omniprésente, largement appliquées par les spécialistes de l'apprentissage automatique dans des domaines variés, tels que le traitement d'informations [32], la reconnaissance de l'écriture manuscrite [21], la reconnaissance faciale, les applications médicales comme le traitement d'images par rayons X [33] et la classification de la fréquence cardiaque [34]. On peut également citer leurs applications dans la reconnaissance de cibles radar [35], les prévisions financières [36, 39, 38], l'analyse de bases de données [13] et la bioinformatique [18].

La méthode du support est une technique utilisée pour résoudre des programmes linéaires(PLs) avec des variables bornées. Elle est particulièrement utile lorsque le problème primal (le PL original) comporte un grand nombre de variables, et utilisent un critère d'arrêt de suboptimalité. Le principe de cette méthode est simple : partant d'une solution réalisable de support initiale,

chaque itération consiste à trouver une direction d'amélioration et un pas maximal le long de cette direction en améliorant la valeur de la fonction objectif, et tout en veillant à ne pas sortir du domaine réalisable déterminé par les contraintes du problème.

La régularisation en machine learning est une étape cruciale dans le processus d'apprentissage des SVM, car elle permet de contrôler la complexité du modèle et d'éviter le sur-apprentissage, garantissant ainsi une meilleure généralisation sur de nouvelles données. Dans ce mémoire, nous allons résoudre les problèmes de SVM régularisés en utilisant la norme L_2 et une fonction de perte de norme L_1. La recherche des paramètres de l'hyperplan séparateur optimal revient alors à résoudre un problème quadratique convexe (PQC) à variables bornées.

Nous proposerons un algorithme itératif et simple à implémenter en Python. Pour illustrer les performances de notre approche par rapport à la méthode SMO (Sequential Minimal Optimization) [4], nous effectuerons des expérimentations numériques sur quelques bases d'apprentissage. Les critères de comparaison utilisés seront le temps d'exécution, le nombre d'itérations, le nombre de vecteurs support et la précision (accuracy).

Ce mémoire est constitué d'une introduction générale, de quatre chapitres et d'une conclusion générale.

Le premier chapitre, présente les concepts fondamentaux de l'optimisation, y compris la forme quadratique, la convexité, ainsi que les conditions d'optimalité associées à ces problèmes. Il présente également la dualité lagrangienne et ses applications en optimisation quadratique convexe.

Le deuxième chapitre porte sur les problèmes de classification par SVMs, en commençant par les bases de l'apprentissage statistique et l'objectif des SVMs. Il explore ensuite les fondements mathématiques des SVMs, y compris la formulation du problème de classification, les fonctions noyaux, et les propriétés associées. Les différentes approches pour les SVM linéaires et non linéaires sont également discutées.

Le troisième chapitre se concentre sur la méthode de support pour la minimisation des problèmes quadratiques convexes à variables bornées. Nous les reformulons de manière pratique afin de développer un algorithme de support qui cherche effectivement une solution optimale, illustré par un exemple numérique.

Le quatrième chapitre est consacré à l'implémentation de la méthode de support sous le langage Python, ainsi qu'à son application aux problèmes des machines à vecteurs de support (SVMs). Afin de comparer son efficacité face à la méthode SMO, nous effectuons des expérimentations numériques des septs benchmarks de la base UCI.

Chapitre 1

Introduction à la programmation quadratique

L'optimisation quadratique est l'une des théories de la programmation mathématique les plus couramment utilisées pour modéliser des problèmes pratiques. C'est une théorie autonome de l'optimisation non linéaire, dans laquelle on cherche à minimiser (ou maximiser) une forme quadratique sous des contraintes linéaires.

Ce chapitre introduit les concepts fondamentaux de l'optimisation, y compris la forme quadratique, la convexité, ainsi que les conditions d'optimalité associées à ces problèmes. Pour finir, nous représentons les conditions d'optimalité pour un problème quadratique convexe et de la dualité lagrangienne.

1.1 Formes quadratiques

Définition 1.1. Soit F une fonction de n variable $x_1, x_2, \ldots, x_n$ définie de $\mathbb{R}^n$ dans $\mathbb{R}$. La fonction F est dite forme quadratique si et seulement si elle s'écrit sous la forme suivante :

$$F(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i x_j + \sum_{j=1}^{n} c_j x_j = x'Ax + c'x \tag{1.1}$$

où $x = \Big(x_1, x_2, \ldots, x_n\Big)'$ et c un n-vecteur colonne ; $A = \Big(a_{ij}, 1 \leq i, j \leq n\Big)$ est une matrice carrée d'ordre n.

- Le symbole $(')$ est l'opérateur de transposition vectorielle et matricielle.
- Pour $i \neq j$, le coefficient du terme $x_i x_j$ s'écrit $a_{ij} + a_{ji}$ dans ce cas la matrice A peut être supposée symétrique.
- Noté bien qu'en définissant de nouveaux coefficients

  $$d_{ij} = d_{ji} = \frac{a_{ij} + a_{ji}}{2}, \qquad 1 \leq i, j \leq n,$$

  on obtient une nouvelle matrice $D = \Big(d_{ij}, 1 \leq i, j \leq n\Big)$ symétrique et par conséquent la forme quadratique $F(x)$ reste inchangée :

  $$F(x) = x'Ax + c'x = x'Dx + c'x \qquad \forall x \in \mathbb{R}^n$$

1.1.1 Représentation matricielle d'une forme quadratique

Définition 1.2. Soient $x = (x_1, x_2, \ldots, x_n)'$ et $D = (d_{ij},\ 1 \leq i \leq n,\ 1 \leq j \leq n)$
Pour n = 2 et d'après la formule (1.1), on a :

$$\begin{aligned} F(x) &= \sum_{i=1}^{2}\sum_{j=1}^{2} a_{ij}x_ix_j + \sum_{j=1}^{2} c_jx_j \\ &= a_{11}x_1^2 + a_{12}x_1x_2 + a_{22}x_2^2 + a_{21}x_2x_1 + c_1x_1 + c_2x_2 \\ &= x_1(a_{11}x_1 + a_{12}x_2) + x_2(a_{21}x_1 + a_{22}x_2) + c_1x_1 + c_2x_2 \\ &= \begin{pmatrix} x_1 & x_2 \end{pmatrix}\begin{pmatrix} a_{11}x_1 + a_{12}x_2 \\ a_{21}x_1 + a_{22}x_2 \end{pmatrix} + \begin{pmatrix} c_1 & c_2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ &= \begin{pmatrix} x_1 & x_2 \end{pmatrix}\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} c_1 & c_2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ &= x'Dx + c'x. \end{aligned}$$

Exemple 1.1. Considérons la forme quadratique suivante :

$$F(x) = 2x_1^2 + x_1x_2 + x_2^2 - 12x_1 - 10x_2 \tag{1.2}$$

(1.3)

La matrice symétrique associée

$$D = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \quad \text{et} \quad c = \begin{pmatrix} -12 \\ -10 \end{pmatrix} \tag{1.4}$$

1.1.2 Gradient d'une forme quadratique

Définition 1.3. Soit $F : \mathbb{R}^n \to \mathbb{R}$ une fonction continûment différentiable. Son gradient au point x est défini par :[22]

$$\nabla F(x) = \begin{pmatrix} \frac{\partial F}{\partial x_1} \\ \frac{\partial F}{\partial x_2} \\ \vdots \\ \frac{\partial F}{\partial x_n} \end{pmatrix} = 2Dx + c. \tag{1.5}$$

où $\frac{\partial F}{\partial x_i}$ est la dérivée partielle de F par rapport à x_i.

Définition 1.4. Soit $F : \mathbb{R}^n \to \mathbb{R}$. une fonction réelle de classe C^2, $F : \mathbb{R}^n \to \mathbb{R}$. Le Hessien de la fonction F est défini par :

$$\nabla^2 F(x) = \left(\nabla\frac{\partial F(x)}{\partial x_1}, \nabla\frac{\partial F(x)}{\partial x_2}, ..., \nabla\frac{\partial F(x)}{\partial x_n}\right) = \begin{pmatrix} \frac{\partial^2 F(x)}{\partial x_1^2} & \frac{\partial^2 F(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 F(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 F(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 F(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 F(x)}{\partial x_2 \partial x_n} \\ \vdots & & \ddots & \\ \frac{\partial^2 F(x)}{\partial x_n \partial x_1} & \frac{\partial^2 F(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 F(x)}{\partial x_n^2} \end{pmatrix}$$

Où $\frac{\partial^2 F}{\partial x_i^2}$ est la dérivée partielle d'ordre 2 de F.

1.2 Caractérisation d'une forme quadratique

Consididérons la forme quadratique (1.1) ,où D est une matrice symétrique d'ordre n.

1.2.1 Matrice définie, semi-définie et non définie

.

Définition 1.5. $F(x)$ est dite définie positive si :

$$x'Dx > 0, \forall x \in \mathbb{R}^n, x \neq 0. \tag{1.6}$$

Définition 1.6. $F(x)$ est dite semi-définie positive si :

$$x'Dx \geq 0, \forall x \in \mathbb{R}^n. \tag{1.7}$$

Définition 1.7. $F(x)$ est dite définie négative si :

$$x'Dx < 0, \forall x \in \mathbb{R}^n, x \neq 0. \tag{1.8}$$

Définition 1.8. $F(x)$ est dite semi-définie négative si :

$$x'Dx \leq 0, \forall x \in \mathbb{R}^n. \tag{1.9}$$

Définition 1.9. La forme quadratique $F(x) = x'Dx + c'x$ est dite définie positive (resp. semi-définie positive) si sa matrice associée D est définie positive (resp. semi-définie positive).

Remarque 1.2.1. Si la forme quadratique $x'Dx$ est positive pour certaines valeurs de x et négative pour d'autres, alors D est dite *indéfinie*.

1.2.2 Critère de SYLVESTER

Garce au **Critère du Sylvester** on peut caractériser une forme quadratique définie positive ou semi-définie positive.
Pour cela, considérons la matrice symétrique suivante :

$$D = \begin{pmatrix} d_{11} & d_{12} & ... & d_{1n} \\ d_{21} & d_{22} & ... & d_{2n} \\ \vdots & & \ddots & \\ d_{n1} & d_{n2} & ... & d_{nn} \end{pmatrix}$$

Le mineur de la matrice D, formé des lignes $i_1, i_2, ..., i_p$, et des colonnes $j_1, j_2, ..., i_p$, sera noté comme suit :

$$D\begin{pmatrix} i_1, i_2, ..., i_p \\ j_1, j_2, ..., i_p \end{pmatrix} = \begin{vmatrix} d_{i_1 j_1} & d_{i_1 j_2} & ... & d_{i_1 j_p} \\ d_{i_2 j_1} & d_{i_2 j_2} & ... & d_{i_2 j_P} \\ \vdots & & \ddots & \\ d_{i_p j_1} & d_{i_p j_{\acute{e}}} & ... & d_{i_p j_p} \end{vmatrix}$$

Ce mineur est dit principal si $i_1 = j_1, i_2 = j_2, ..., i_p = j_p$, c'est-à-dire, s'il est forme de lignes et de colonnes portant mêmes numéros. Les mineurs suivants :

$$D_1 = d_{11}, \quad D_2 = \begin{vmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{vmatrix}, \ldots, \quad D_n = \begin{vmatrix} d_{11} & d_{12} & ... & d_{1n} \\ d_{21} & d_{22} & ... & d_{2n} \\ \vdots & & \ddots & \\ d_{n1} & d_{n2} & ... & d_{nn} \end{vmatrix} = D,$$

sont appelés mineurs principaux successifs. le critère de Sylvester est décrit par le théorème suivant :

Théorème 1.1. *[23] Soit D une matrice symétrique :*

(i) *Pour que D soit définie positive ($D > 0$), il est nécessaire et suffisant que tous ses mineurs principaux successifs soient positifs :*

$$D_1 > 0, \quad D_2 > 0, \quad ..., \quad D_n > 0. \tag{1.10}$$

(ii) *La matrice D est semi-définie positive ($D \geq 0$), si et seulement si tous ses mineurs principaux sont positifs ou nuls.*

Les formes quadratiques sont aussi reliées aux valeurs propres de leurs matrices associées :

Théorème 1.2. *Soit D une matrice symétrique d'ordre n. On note par $\{\lambda_i\}_{=1,...,n}$ ses valeurs propres réelles. On a les équivalences suivantes :*

1. $D \geq 0 \iff \lambda_i \geq 0, i = 1, ..., n.$
2. $D > 0 \iff \lambda_i > 0, i = 1, ..., n.$
3. *D est dite indéfinie si et seulement si certains λ_i sont positifs et d'autres négatives.*

1.3 Notion de convexité

Dans les problèmes de minimisation, avec ou sans contraintes, la notion de convexité joue un rôle très important. En effet, pour la plupart des algorithmes, la convergence vers un minimum global ne pourra être démontrée qu'avec l'hypothèse de convexité.

1.3.1 Ensemble convexe

Définition 1.10. [7] Une ensemble S dans $\mathbb{R}^n$ est dit convexe si :

$$\lambda x + (1 - \lambda)y \in S, \quad \forall x, y \in S, \quad \lambda \in [0, 1]. \tag{1.11}$$

Ceci revient à dire que S est convexe si pour tout x et y de S, le segment de droite [x,y] est tout entier contenu dans S comme le montre la figure suivante :

1.3.2 Fonction convexe

Définition 1.11. [24] Soit F une fonction réelle définie sur l'ensemble convexe S. F est dite convexe si et seulement si elle vérifie la relation suivante :

$$F(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda F(x_1) + (1 - \lambda)F(x_2), \quad \forall (x_1, x_2) \in S^2, \ \forall \lambda \in [0, 1]. \tag{1.12}$$

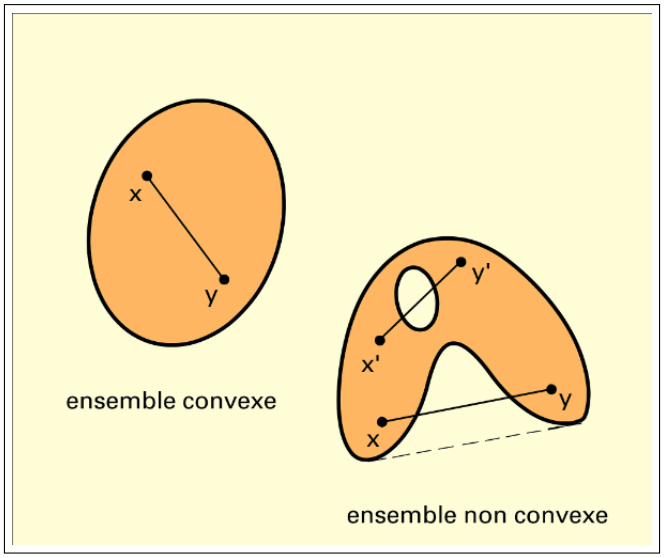

FIGURE 1.1 – Ensemble convexe et non convexe

Définition 1.12. Une fonction convexe $F(x)$, $x \in S$, est dite strictement convexe si l'inégalité (1.12) est stricte pour $x_1 \neq x_2$ et $\lambda \in]0, 1[$

Pour tous les problèmes de programmation convexe, on a les propriétés suivantes [28] :

Propriété 1.1. Soit F une fonction convexe définie sur un ensemble convexe $C \subset \mathbb{R}^n$. Alors, l'ensemble des points où F atteint son minimum est convexe.

Propriété 1.2. Soit $F : C \subset \mathbb{R}^n \to \mathbb{R}$ une fonction convexe. Alors, tout minimum local est un minimum global.

Propriété 1.3. Tout problème strictement convexe admet au plus une solution.

Propriété 1.4. Si la fonction F est strictement convexe, alors son minimum global, lorsqu'il existe, est atteint en un seul point x^*.

Les figures suivantes illustrent respectivement une fonction convexe et non convexe :

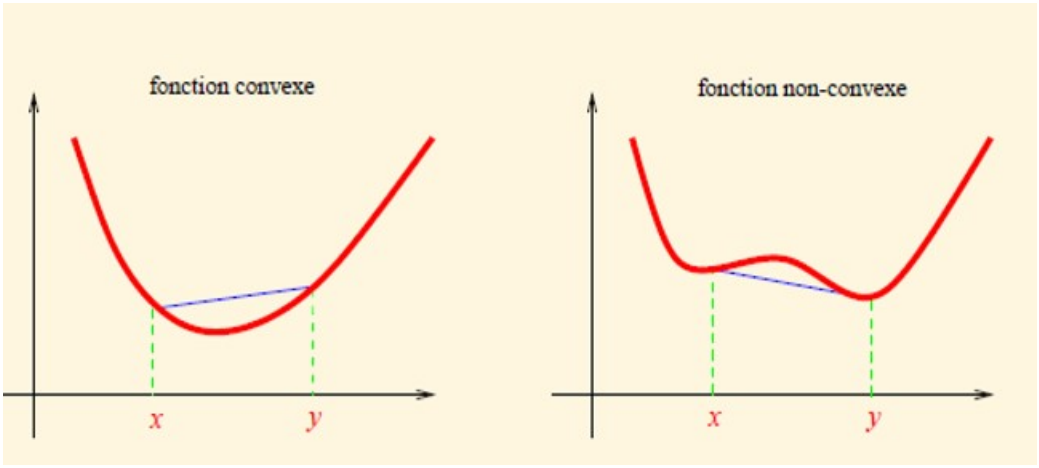

FIGURE 1.2 – Fonction convexe (à gauche) et fonction non convexe (à droite).

1.4 Conditions d'optimalité pour un problème non linéaire

1.4.1 Position du problème

Définition 1.13. Soit F et g_i les fonctions réelles définies et dérivables sur $\mathbb{R}^n$.
Un problème d'optimisation non-linéaire consiste à minimiser (maximiser) une fonction objectif F sous des contraintes g_i, $i = 1 \dots m$ de type égalité et inégalité :

$$\begin{cases} \min_x F(x) \\ \\ \quad g_i(x) = 0 \quad i \in \mathcal{E}, \\ \\ \quad g_i(x) \leq 0 \quad i \in \mathcal{I}, \\ \end{cases} \tag{1.13}$$

où

— $x = (x_1, x_2, ..., x_n)' \in \mathbb{R}^n$ est le vecteur des variables de décision,

— F et g_i sont des fonctions réelles définies et dérivables sur $\mathbb{R}^n$ et au moins l'une d'elles est non linéaire.

— $\mathcal{E} = \{1, \dots, m_e\}$: l'ensemble des indices des contraintes d'égalités,

— $\mathcal{I} = \{m_e, \dots, m\}$: l'ensemble des indices des contraintes d'inégalités,

— m_e et m sont des entiers, tels que $0 \leq m_e \leq m$.

Remarque 1.4.1.

— Si $m = 0$, alors (1.13) est un problème d'optimisation sans contraintes.

— Si $m_e = m \neq 0$, alors (1.13) est un problème d'optimisation avec contraintes d'égalités,

— Si $g_i(x)$ $i = 1, \dots, m$ sont linéaires, alors (1.13) est un problème d'optimisation avec contraintes linéaires.

— Si la fonction objectif $F(x)$ est quadratique et les contraintes sont linéaires, alors (1.13) est un problème d'optimisation quadratique,

— Le problème (1.13) peut être écrit sous la forme compacte suivante :

$$\min_{x \in S} F(x), \tag{1.14}$$

où l'ensemble des solutions réalisables S est défini par :

$$S = \{x \mid g_i(x) = 0,\ i \in \mathcal{E}\ ;\ g_i(x) \leq 0,\ i \in \mathcal{I}\}.$$

Définition 1.14. (minimum local et minimum global)[23]
Le point $x^* \in S$ est dit :

- minimum local du problème (1.13) si : $F(x^*) \leq F(x)$, $\forall x \in S \cap v(x^*, \varepsilon)$, où $v(x^*, \varepsilon) = \{x \in \mathbb{R}^n \ ||x - x^*|| \leq \varepsilon\}$ est un voisinage de x^*.

- minimum global du problème (1.13) si : $F(x^*) \leq F(x)$, $\forall x \in S$.

Définition 1.15. (contrainte active, inactive et violée)
Soit $g_i(x)$, $i \in \mathcal{I}$, la $i^{\text{ème}}$ contrainte d'inégalité du problème (1.13) .

- si $g_i(x) = 0$, alors elle est dite contrainte active au point x,
- si $g_i(x) < 0$, alors elle est dite contrainte inactive,
- si $g_i(x) > 0$, alors elle est dite contrainte violée.

On note :

$I_a(x) = \{i \mid g_i(x) = 0,\ \ i \in I\}$ l'ensemble des indices des contraintes actives au point x avec $I = \varepsilon \cup \mathcal{I}$.

Définition 1.16. (direction admissible)[23] Soit $x \in S, 0 \neq d \in \mathbb{R}^n$. S'il existe un réel $\alpha > 0$,tel que :

$$x + \delta d \in S,\ \forall \delta \in [0, \alpha],$$

alors d est dite direction réalisable (ou admissible) au point $x \in S$.

Définition 1.17. (la direction admissible linéaire)[23] la direction d est dite admissible linéaire au point $x \in S$, si elle vérifie les relations suivantes :

$$d'\nabla g_i(x) = 0,\ \ i \in \mathcal{E},$$
$$d'\nabla g_i(x) \leq 0,\ i \in \mathcal{I}.$$

L'ensemble des directions admissibles linéaires au point $x \in S$ est :

$$\mathcal{F}(x) = \left\{ d \;\middle|\; \begin{array}{ll} d'\nabla g_i(x) = 0, & i \in \mathcal{E}, \\ d'\nabla g_i(x) \leq 0, & i \in \mathcal{I}, \end{array} \right\} \tag{1.15}$$

Définition 1.18. (directions admissibles séquentielles)[23] Soient $x \in S$, $d \in \mathbb{R}^n$. S'il existe deux suites d_k et δ_k, $k = 1, 2, ..., n$ telles que : $x + \delta_k d_k \in S$, $\forall k$ et $d_k \to d$, $\delta_k \to 0$, alors la direction finale d est appelée direction admissible séquentielle de S en x. L'ensemble de toutes les directions admissibles séquentielles de S en x est :

$$\mathcal{T}(x) = \left\{ d \;\middle|\; \begin{array}{ll} x + \delta_k d_k \in S, & \forall k, \\ d_k \to d, & \delta_k \to 0 \end{array} \right\} \tag{1.16}$$

1.4.2 Condition de minimalite de Karush-Kuhn-Tucker

Condition du premier ordre [23]

Lemme 1.3. *Si $x^* \in S$ est un minimum local du problème* (1.13) *est F est de classe C^1, alors pour toute direction admissible d en x^*, nous avons :*

$$d'\nabla F(x^*) \geq 0 \tag{1.17}$$

Définition 1.19. (Condition de qualification des contraintes QC)[23] La condition de qualification des contraintes (QC) est une supposition faite par Kuhn et Tucker (1951), où $\mathcal{T}(x) = \mathcal{F}(x)$ qui signifie que $\forall d \in \mathcal{F}(x)$ il existe deux suites d_k et δ_k $k = 1, 2, ...,$ telles que $x + \delta_k d_k \in S$, $\forall k$ et $d_k \to d$, $\delta_k \to 0$.

Lemme 1.4. *Pour que la condition de qualification des contraintes soit vérifiée, il suffit que l'une des deux conditions ci-dessous soit vérifiée :*

- $g_i(x)$, $i \in I_a(x)$ *sont linéaires*
- $\nabla g_i(x)$, $i \in I_a(x)$ *sont linéairement indépendants*

Fonction de Lagrange [22]

La fonction $\mathcal{L}(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$ est appelée fonction de Lagrange associée au problème de minimisation , où $\lambda = (\lambda_1, \lambda_2, ..., \lambda_m) \in \mathbb{R}^m$ est le vecteur multiplicateur de Lagrange associé aux contraintes du problème (1.13).

Théorème 1.5. ***(Lemme de Motzkin)****[26] Soient $A_1 \in \mathbb{R}^{m_1 \times n}$, $A_2 \in \mathbb{R}^{m_2 \times n}$ et $c \in \mathbb{R}^n$, alors exactement un des deux systèmes suivants admet une solution :*

a) $A_1 x = 0$, $A_2 x \leq 0$, $c^{'} x > 0$

b) $A_1^{'} y_1 + A_2^{'} y_2 = c$, $y_2 \geq 0$

Lemme 1.6. ***(Farkas)****[26]*

i) $G = \{d \mid d^{'} \nabla f(x^*) < 0, d^{'} \nabla g_i(x^*) = 0 : i \in \mathcal{E}, d^{'} \nabla g_i(x^*) \leq 0 : i \in \mathcal{I}\}$

est un ensemble vide si et seulement s'il existe des nombres réels λ_i, $i \in \mathcal{E}$, et des réels non négatifs λ_i, $i \in \mathcal{I}$, tels que :

ii) $-\nabla f(x^*) = \sum_{i \in \mathcal{E}} \lambda_i \nabla g_i(x^*) + \sum_{i \in I} \lambda_i \nabla g_i(x^*)$

Remarque 1.4.2. Le lemme de Farkas découle directement du lemme de Motzkin. En effet, en faisant les transformations suivantes sur (i) et (ii) du lemme de Farkas :
$x = d$, $c = -\nabla f(x^*)$, $A_1 = (\nabla g_1(x^*), ..., \nabla g_{m_e}(x^*))^{'}$, $A_2 = (\nabla g_{m_e+1}(x^*), ..., \nabla g_m(x^*))^{'}$, $y_1 = \lambda(\mathcal{E})$, $y_2 = \lambda(\mathcal{I})$ on retombe sur les deux systèmes (a) et (b) du lemme de Motzkin.

Théorème 1.7. *Si $f(x)$ et $g_i(x)$, $i = 1..m$ sont différentiables en $x^* \in S$ qui est un minimum local du problème* (1.13) *alors :*

$$d^{'} \nabla f(x^*) \geq 0, \qquad \forall d \in \mathcal{T}(x^*)$$

Théorème 1.8. ***(Karush-Kuhn-Tucker)****[14] Si x^* est un minimum local de F sur S et si la condition de qualification des contraintes est vérifiée, alors il existe un vecteur $\lambda^* = (\lambda_1^*, ..., \lambda_m^*)^{'}$, tel que les conditions ci-dessous sont vérifiées en (x^*, λ^*)*

1) $\nabla F(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) = 0$

2) $g_i(x^*) = 0, \forall i \in \mathcal{E}$

3) $g_i(x^*) \leq 0, \forall i \in \mathcal{I}$

4) $\lambda_i^* \geq 0, \forall i \in \mathcal{I}$

5) $\lambda_i^* g_i(x^*) = 0, \forall i \in \mathcal{I}$

Preuve :

- Vérification de la condition de stationnarité (1) et de non-négative des λ_i associés aux contraintes d'inégalités :

Soit $d \in \mathcal{T}(x^*)$ à partir de la condition QC on a $d \in \mathcal{F}(x^*)$, et soit x^* un minimum local de F alors d'après le théorème **(8)** $d^{'}\nabla f(x^*) \geq 0$, ce qui fait que le système ci-dessous n'admet pas de solution :

$$d^{'}\nabla f(x^*) < 0, \qquad d^{'}\nabla g_i(x^*) = 0 : i \in \mathcal{E}, \qquad d^{'}\nabla g_i(x^*) \leq 0 : i \in I_a(x^*)$$

alors, en vertu du lemme de Farkas il existe des coefficients λ_i, $i = 1, ..., m$ qui vérifient :

$$-\nabla f(x^*) = \sum_{i \in \mathcal{E}} \lambda_i^* \nabla g_i(x^*) + \sum_{i \in \mathcal{I}} \lambda_i^* \nabla g_i(x^*) \qquad \text{où } \lambda_i^* \in \mathbb{R} \text{ et } \lambda_i^* \geq 0 \ (i \in I_a(x^*))$$

.
En posant $\lambda_i^* = 0$, $\forall i \in \mathcal{I} \setminus I_a(x^*)$ on aura alors :

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) = 0 \qquad \text{et} \qquad \lambda_i^* \geq 0, \forall i \in \mathcal{I}$$

- Vérification de la condition de complémentarité (5) :

On remarque que dans les deux cas où lorsque on a : $i \in I_a(x^*)$, $g_i(x^*) = 0$ et $\lambda_i^* \geq 0$ ou bien $i \in I \setminus I_a(x^*)$, $g_i(x^*) < 0$ et $\lambda_i^* = 0$ on aura toujours $\lambda_i^* g_i(x^*) = 0$ ce qui vérifie la condition (5)

- Vérification des conditions de faisabilité (2) et (3) :

x^* est un minimum local alors c'est un point réalisable ce qui vérifie les conditions (2) et (3).

Condition d'optimalité du second ordre[23]

On suppose que les fonctions $f(x)$ et $g_i(x)$, $i = 1..m$ sont deux fois continûment différentiables et on définit un ensemble de directions admissibles $\mathcal{K}$, tel que :

$$\mathcal{K}(x^*, \lambda^*) = \{d \neq 0 \mid d^{'}\nabla g_i(x^*) = 0 : \ i \in \mathcal{I}_{a_+}(x^*) \ ; \ \ d^{'}\nabla g_i(x^*) \leq 0 : \ i \in \mathcal{I}_a(x^*) \setminus \mathcal{I}_{a_+}(x^*)\},$$

où, $\mathcal{I}_{a_+}(x^*)$ est obtenu en supprimant de $\mathcal{I}_a(x^*)$ les indices pour lesquels $\lambda_i^* = 0 : i \in \mathcal{I}_a(x^*)$.

Théorème 1.9. ***(condition nécessaire)***
Soit x^* un minimum local du problème (1.13)*. Si les conditions de qualification des contraintes sont vérifiées alors on a :*

$$d^{'}\nabla^2_{xx}\mathcal{L}(x^*, \lambda^*)d \geq 0 \ \ , \forall d \in \mathcal{K}(x^*, \lambda^*)$$

Théorème 1.10. ***(condition suffisante)***
soit x^* un point de KKT du problème (1.13)*. Si :*

$$d^{'}\nabla^2_{xx}\mathcal{L}(x^*,\lambda^*)d > 0 \ \ ,\forall d \in \mathcal{K}(x^*,\lambda^*),$$

alors x^* est un minimum local strict du problème (1.13)*.*

1.5 Dualité en optimisation convexe

Le concept de dualité consiste à trouver un problème dit dual qui n'est rien d'autre qu'une reformulation équivalente d'un problème initial appelé problème primal, et ce dans le but est de trouver un problème plus pratique à résoudre.

Définition 1.20. (Problème dual)

La forme du problème dual est :

$$\begin{cases} \max \mathcal{L}(y,\lambda), \\ \\ \quad \nabla_y \mathcal{L}(y,\lambda) = 0, \\ \\ \quad \lambda_i \geq 0, \qquad i \in I. \end{cases} \tag{1.18}$$

Théorème 1.11. ***(dualité faible)***
Soit x (resp.(y,λ)) une solution réalisable du problème primal (1.13) *(resp.dual* (2.7)*) alors :* $f(x) \geq \mathcal{L}(y,\lambda), \forall x \in S, \forall (y,\lambda) \in \mathbb{V}$, $V = \{(y,\lambda) : \nabla L(y,\lambda) = 0, \lambda_i \geq 0, o\grave{u} i \in I\}$.

Théorème 1.12. ***(dualité forte)***
Soit x^* un minimum du problème (1.13)*, si $f(x)$ et $g_i(x)$, $i \in I$ sont continûment différentiables et si la condition de qualification des contraintes est vérifiée, alors (x^*,λ^*) est une solution du problème dual* (2.7)*, de plus le minimum du primal est égal au maximum du dual :* $f(x^*) = \mathcal{L}(x^*,\lambda^*)$.

Remarque 1.5.1. Si le problème primal est non borné, alors son dual est irréalisable et vice-versa.

1.6 Programmation quadratique convexe

Définition 1.21. (Programmation quadratique convexe)
On appelle problème de programmation quadratique convexe (PQC) tout problème de programmation non-linéaire, qui consiste à minimiser une fonction quadratique convexe $F(x)$ sur un ensemble convexe S défini par des contraintes linéaires. La forme d'un (PQC) est :

$$\begin{cases} \min \quad F(x) = \frac{1}{2}x^{'}Dx + c^{'}x \\ \\ \qquad a_i^{'}x - b_i = 0, \quad i \in \mathcal{E}, \\ \qquad a_i^{'}x - b_i \leq 0, \quad i \in \mathcal{I}, \end{cases} \tag{1.19}$$

où : $I = \mathcal{E} \cup \mathcal{I} = \{1, 2, ..., m\}$ est l'ensemble des indices des contraintes, $D = D^{'}$ une matrice carrée d'ordre n semi-définie positive, x, c et a_i, $i \in \{1, 2, ..., m\}$ sont des n-vecteurs, $b = (b_i,\ i \in \mathcal{E} \cup \mathcal{I})$ est un m-vecteur.

Propriétés des PQC

Propriété 1.5. Pour un problème de programmation quadratique convexe sans contrainte la première condition de KKT (stationnarité) constitue une condition à la fois nécessaire et suffisante de minimalité globale.

Théorème 1.13. *Si x^* est un minimum local d'un problème convexe, alors il est aussi un minimum global, de plus l'ensemble des points où la fonction objectif atteint son minimum est convexe.*

Théorème 1.14. *Pour un problème convexe, si les conditions de KKT sont vérifiées au point x^*, alors celui-ci est un minimum global.*

1.6.1 Dual d'un problème quadratique convexe :

Notons par $A = \left(a_1^{'} \cdots a_m^{'}\right)$ la matrice associée aux contraintes du problème (1.19). La fonction de Lagrange associée au problème primal (1.19) est la suivante [25] :

$$\mathcal{L}(x, \lambda) = \frac{1}{2} x^{'} D x + c^{'} x + \lambda^{'}(Ax - b)$$

Les conditions nécessaires et suffisantes d'optimalité d'un point (x, λ) pour (1.19) sont :

$$\frac{\partial \mathcal{L}}{\partial x} = Dx + c + A^{'}\lambda = 0, \tag{1.20}$$

$$a_i^{'} x - b_i = 0,\ \ i \in \mathcal{E}, \quad a_i^{'} x - b_i \leq 0,\ \ i \in \mathcal{I}, \tag{1.21}$$

$$\lambda_i (a_i^{'} x - b_i) = 0,\ \ \lambda_i \geq 0,\ \ i \in \mathcal{I}. \tag{1.22}$$

Le dual associé à (1.19) est :

$$\begin{cases} \max \mathcal{L}(x, \lambda), \\ \frac{\partial \mathcal{L}(x,\lambda)}{\partial x} = 0, \\ \lambda \geq 0,\ \ i \in \mathcal{I}. \end{cases}$$

Multiplions la relation (1.20) par $x^{'}$ on aura :

$$x^{'} D x + c^{'} x + x^{'} A^{'} \lambda = 0 \Leftrightarrow x^{'} D x + c^{'} x + \lambda^{'} A x = 0$$

La fonction de Lagrange aura finalement la forme suivante :

$$\begin{aligned} \mathcal{L}(x, \lambda) &= \frac{1}{2} x^{'} D x + c^{'} x + \lambda^{'}(Ax - b) - (x^{'} D x + c^{'} x + \lambda^{'} A x) \\ &= -\frac{1}{2} x^{'} D x - \lambda^{'} b. \end{aligned}$$

Le problème dual associé à (1.19) aura la forme finale suivante :

$$\begin{cases} \max \mathcal{L}(x,\lambda) = -\frac{1}{2}x^{'}Dx - \lambda^{'}b, \\ \\ Dx + c + A^{'}\lambda = 0, \\ \\ \lambda_i \geq 0, \;\; i \in \mathcal{I}, \quad \lambda_i \in \mathbb{R}, \;\; i \in \mathcal{E}. \end{cases}$$

1.7 Domaines d'application

L'application de la programmation quadratique convexe est de plus en plus en expansion croissante et on trouve beaucoup d'applications dans plusieurs domaines, on peut citer :

1. L'analyse structurelle [29].
2. Les machines à vecteurs de supports(SVM) [9, 15].
3. Le contrôle optimal [29].
4. L'économie mathématique [30].

1.8 Conclusion

Dans ce chapitre, nous avons présenté les notions et les propriétés des formes quadratiques. En plus, nous avons vu la notion de la convexité et les propriétés des fonctions convexes. Ensuite, nous avons énuméré les conditions d'optimalité de KKT pour les problèmes d'optimisation non linéaire avec contraintes. Enfin, nous avons achevé ce chapitre par la présentation de la programmation quadratique convexe et la théorie de la dualité.

Chapitre 2

Classification par les Machines à Vecteurs Supports

Parmi les méthodes à noyaux, inspirées de la théorie statistique de l'apprentissage de Vladimir Vapnik, les Machines à Vecteurs de Support (SVMs) constituent la forme la plus connue. SVM est une méthode de classification binaire par apprentissage supervisé, elle fut introduite par Vapnik en 1995[9]. Cette méthode est donc une alternative récente pour la classification. Elle repose sur l'existence d'un classificateur linéaire dans un espace approprié.

Puisque c'est un problème de classification à deux classes, cette méthode fait appel à un jeu de données d'apprentissage pour apprendre les paramètres du modèle. Elle est basée sur l'utilisation de fonctions dites noyau (kernel) qui permettent une séparation optimale des données. Dans la présentation des principes de fonctionnements, nous schématiserons les données par des « points » dans un plan.

Dans ce chapitre, nous allons présenter les machines à vecteures support, leur origine théorique, leur principe et notion de base, leurs défférentes formes, ainsi que les avantages et les désavantages des SVM.

2.1 L'objectif de l'apprentissage statistique et SVM

La théorie d'apprentissage statistique étudie les propriétés mathématiques des machines d'apprentissage[17]. Ces propriétés représentent les propriétés de la classe de fonctions ou modèles que peut implémenter la machine. L'apprentissage statistique utilise un nombre limité d'entrées (appelées exemples) d'un système avec les valeurs de leurs sorties pour apprendre une fonction qui décrit la relation fonctionnelle existante, mais non connue, entre les entrées et les sorties du système[8]. On suppose premièrement que les exemples d'apprentissage, appelés aussi exemples d'entrainement, sont générés selon une certaine probabilité inconnue, c'est-à-dire indépendants et identiquement distribués. Ensuite, chaque exemple tracé comme un point dans un espace de dimension ($x_i \in R^p$) pour le cas d'apprentissage supervisé, accompagnés d'étiquettes caractérisant leurs types ou classes d'appartenance. Si ces étiquettes sont dénombrables, on parle de classification sinon on parle de régression. Dans le cas d'une classification binaire cette étiquette $y_i \in \{+1, -1\}$. L'ensemble des exemples et leurs étiquettes correspondantes est appelés ensemble d'apprentissage. Le but est d'apprendre une fonction f qui correspond aux exemples vus et qui prédit les sorties pour les entrées qui n'ont pas encore été vues. Les entrées peuvent être des descriptions d'objets et les sorties la classe des objets donnés en entrée.
Soit : $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ l'ensemble des exemples d'entrainement $x_i \in R^p$ et

$y_i = \pm 1$ f(x) est la fonction apprise par la machine d'apprentissage[18].

$$R_{\text{emp}} = \frac{1}{2n} \sum L(x_i, f(x_i)) \tag{2.1}$$

Avec

$$L = \begin{cases} 1 & \text{si } y_i = f(x_i) \\ 0 & \text{sinon} \end{cases}$$

Où L désigne une fonction de coût.
Pour garantir la fiabilité de f sur des exemples ne participant pas à l'apprentissage, c'est-à-dire qu'ils n'ont jamais été vus auparavant, nous les mesurons sur différents exemples appelés exemples de test et il s'agit d'un test machine. Donc f minimise les erreurs de classification sur les deux ensembles : d'entrainement et de test[18].
Dans le cas des machines à vecteur support, la fonction recherchée est de forme linéaire. Les SVM sont, donc des systèmes d'apprentissage qui utilisent un espace d'hypothèses de fonctions linéaires dans un espace de caractéristique à haute dimension[10].

2.2 L'objectif des machines à vecteurs de support (SVM)

L'objectif des machines à vecteurs de support (SVM) dans la classification binaire est de trouver un hyperplan qui sépare de manière optimale les points de données appartenant à différentes classes. Idéalement, cet hyperplan représente la plus grande marge entre les deux classes, indiquée par les points les plus proches de chaque classe. La marge fait référence à la plus grande distance entre l'hyperplan et les points de données les plus proches, formant ainsi une zone vide parallèle à l'hyperplan. Les SVM visent à maximiser cette marge tout en permettant une certaine flexibilité pour traiter des problèmes non linéaires en utilisant des noyaux. Seuls les vecteurs de support, qui sont des points de données critiques pour définir l'hyperplan, sont nécessaires pour construire le modèle une fois entraîné. Les SVM peuvent également être utilisées pour la détection d'anomalies en concevant des SVM spécifiques pour identifier des valeurs aberrantes. En résumé, l'objectif principal des SVM dans la classification binaire est de trouver un hyperplan avec la plus grande marge possible entre les classes, ce qui permet une classification précise et robuste même en présence de valeurs aberrantes[11].

2.3 Fondement mathématique des SVM

Le fondement mathématique des SVM est expliqué dans plusieurs travaux comme ceux de [9, 2, 15, 19].

2.3.1 Formulation du problème de classification

Le but de la tâche de classification est de prévoir la classe y_i d'un p-vecteur x_i en se basant sur les mesures des variables qui l'expliquent avec pour seule information celle contenue dans l'échantillon d'apprentissage S .
Dans le cas de la classification binaire, nous supposons que les données sont des couples $(x_i, y_i) \in X \times Y, i \in I = \{1, 2, ..., n\}$. Xdésigne l'espace des variables explicatives (espace d'entrée) suvent pris dans $\mathbb{R}^p$, $y_i \in Y$={-1,1} est l'ensemble des classes (espace de sortie) et n est la taille de

l'échantillon. L'appartenance d'une observation x_i, à une classe ou à une autre est matérialisée ici par la valeur -1 ou $+1$ son étiquette y_i. L'échantillon d'apprentissage S est ainsi une collection de réalisations indépendantes et identiquement distribuées (i.i.d.) du couple aléatoire (x, y) dont la distribution P est fixe mais inconnue. Cet ensemble est souvent noté par :

$$S = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\} \subseteq (X \times Y)^n$$

Généralement, la classification binaire est accomplie au moyen d'une fonction à valeurs réelles $f : X \subseteq \mathbb{R}^p \to \mathbb{R}$ Toute observation x_i est affectée à la classe qui correspond au signe de $f(x_i)$: si $f(x_i) > 0, x_i$ est affectée à la classe positive $(+1)$, sinon elle sera dans la classe négative (-1). En classification linéaire, la fonction f est linéaire en x_i et elle prend la forme générale suivante :

$$f(x_i) = w' x_i + b$$

.

où $(w, b) \in \mathbb{R}^p \times \mathbb{R}$ sont des paramètres à estimer à partir des données d'apprentissage, appelés respectivement vecteur poids et biais. La règle de décision est donc donnée par $sign(f(x_i))$, appelé classifieur.

2.3.2 Notions de base

2.3.2.1 La marge

La distance d'un point quelconque x_i à l'hyperplan (H) est :

$$d(x_i) = \frac{|w' \cdot x_i + b|}{\|w\|} \tag{2.2}$$

Les points x_i, les plus proches de H sont situés sur H_1 et H_2, qui sont les hyperplans canoniques, ayant donc une distance :

$$d(x_i) = \frac{1}{\|w\|} \tag{2.3}$$

Ainsi, la marge géométrique entre les deux classes est la distance euclidienne entre H_1et H_2, qui vaut :

$$\gamma = \frac{2}{\|w\|} \tag{2.4}$$

Pourquoi maximiser la marge ?

Le fait d'avoir une marge plus large procure plus de sécurité lorsqu'on classe un nouvel exemple. De plus, si l'on trouve le classificateur qui se comporte le mieux vis-à-vis données d'apprentissage, il est clair qu'il sera aussi celui qui permettra au mieux de classer les nouveaux exemples[12]. Dans le schéma figure 2.1, la partie droite nous montre qu'avec un hyperplan optimal, un nouvel exemple reste bien classé alors qu'il tombe dans la marge. On constate sur la partie gauche qu'avec une plus petite marge, l'exemple se voit mal classé.

2.3.2.2 Fonction de noyau

Il est en général beaucoup plus naturel de définir une fonction noyau $K(x_i, x_j)$ qui vérifie la condition $K(x_i, x_j) =< \phi(x_i), \phi(x_j) >$ [20] plutôt que de créer une fonction ϕ de projection

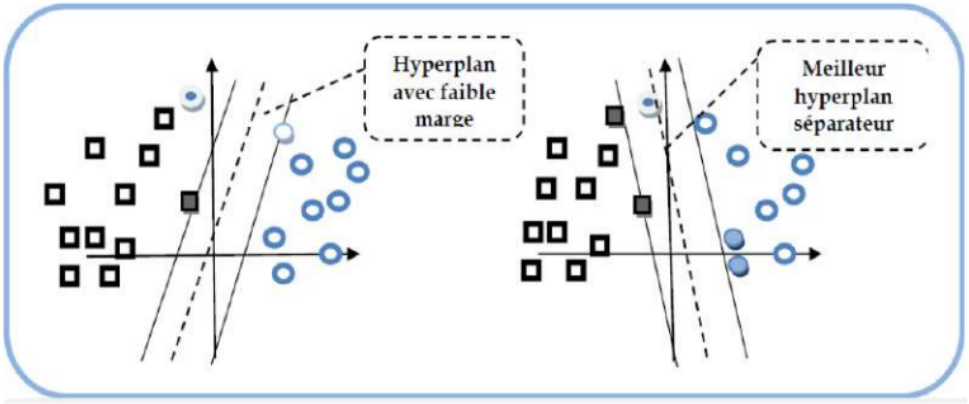

FIGURE 2.1 – Meilleur hyperplan séparateur
[17]

qui permette d'obtenir un espace H suffisamment "flexible" et ensuite de rechercher si une formulation simple de K existe pour que la relation $< \phi(x_i), \phi(x_j) >= K(x_i, x_j)$ soit vérifiée. De plus, des phénomènes de sur-apprentissage peuvent apparaître si la marge correspondante est faible pour les données prises dans cet espace de redescription préconstruit, s'il est trop bien adapté aux données d'entraînement. On ne peut pas pour autant définir une fonction noyau K sans prendre certaines précautions. En particulier, il est indispensable que l'espace de redescription existe et donc que la fonction ϕ existe, même si elle n'est pas facilement exprimable. Dans le cas contraire, le problème de recherche d'un hyperplan optimal n'a pas lieu d'exister.

2.3.3 Noyaux d'usage courant

Une liste non exhaustive de quelques fonctions noyaux couramment utilisées avec les SVM :

- Noyau linéaire : $k(x_i, x_j) = x_i' x_j$ [6]
  Celui-ci correspond au produit scalaire sans transformation. Il traduit donc la forme traditionnelle des algorithmes sans l'usage du kernel trick.
- Noyau polynomial de degré d : $K(x_i, x_j) = (x_i' x_j)^d$ [21]
  Le noyau polynomial élève le produit scalaire à une puissance naturelle d,il permet indirectement d'appliquer le principe de maximisation de la marge aux classifieurs polynômiaux. Si $d = 1$, le noyau devient linéaire.
- Noyau gaussien RBF (Radial Basis functions) : $k(x_i, x_j) = e^{-\sigma(||x_i - x_j||)^2}$ [21]
  Le nombre σ est un réel positif qui représente la largeur de bande du noyau. Les fonctions à base radiale RBF sont définies par le fait qu'elles ne dépendent que de la distance entre leurs arguments :

$$\phi(x, y) = \phi(||x - y||)$$

2.4 Types des SVM

Il existe deux types principaux des SVM :

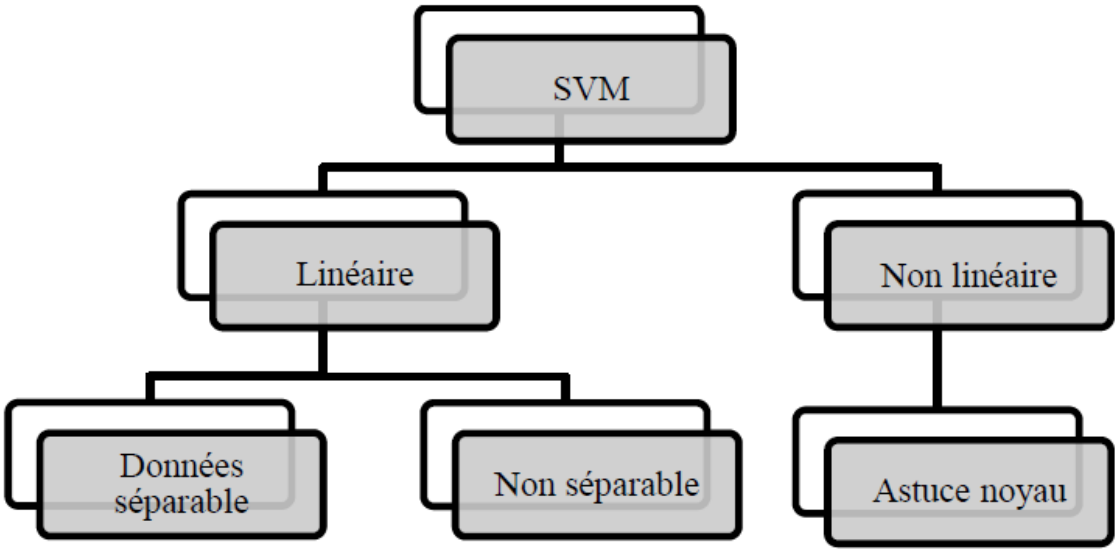

FIGURE 2.2 – Types des SVM.

2.4.1 Classifieur linéaire

Parmi les modèles des SVM linéaire, on constate les cas linéairement séparables et les cas non linéairement séparables. Les SVM linéaire sont utilisé pour les données linéairement séparables, ce qui signifie que si un ensemble de données peut être classé en deux classes à l'aide d'une seule ligne droite, ces données sont appelées données linéairement séparables [12].

2.4.1.1 SVM à marge dure

On se place ici dans le cas où les exemples de l'échantillon S sont linéairement séparables en deux classes par un hyperplan. Même dans ce cas simple, le choix de l'hyperplan séparateur n'est pas évident, car il existe en effet une infinité d'hyperplans séparateurs. Pour résoudre ce problème, Vapnik a montré qu'il existe un unique hyperplan optimal, défini comme étant celui qui maximise la marge entre les sous-échantillons et l'hyperplan séparateur [9].

Il faut trouver les valeurs de w et b qui maximisent la marge afin d'obtenir l'équation de l'hyperplan optimal H :

$$H = w'x + b = 0$$

Étant donné x^+ et x^-, deux points de classes différentes situés sur les frontières positive et négative qui délimitent la marge maximale, on simplifie le problème d'optimisation en considèrant que x^+ et x^- sont respectivement situés sur les hyperplans canoniques $H1$ et $H2$, tels que :

$$H1 : w'x^+ + b = +1$$
$$H1 : w'x^- + b = -1$$

Ainsi,

$$f(x^+) = +1 \quad et \quad f(x^-) = -1.$$

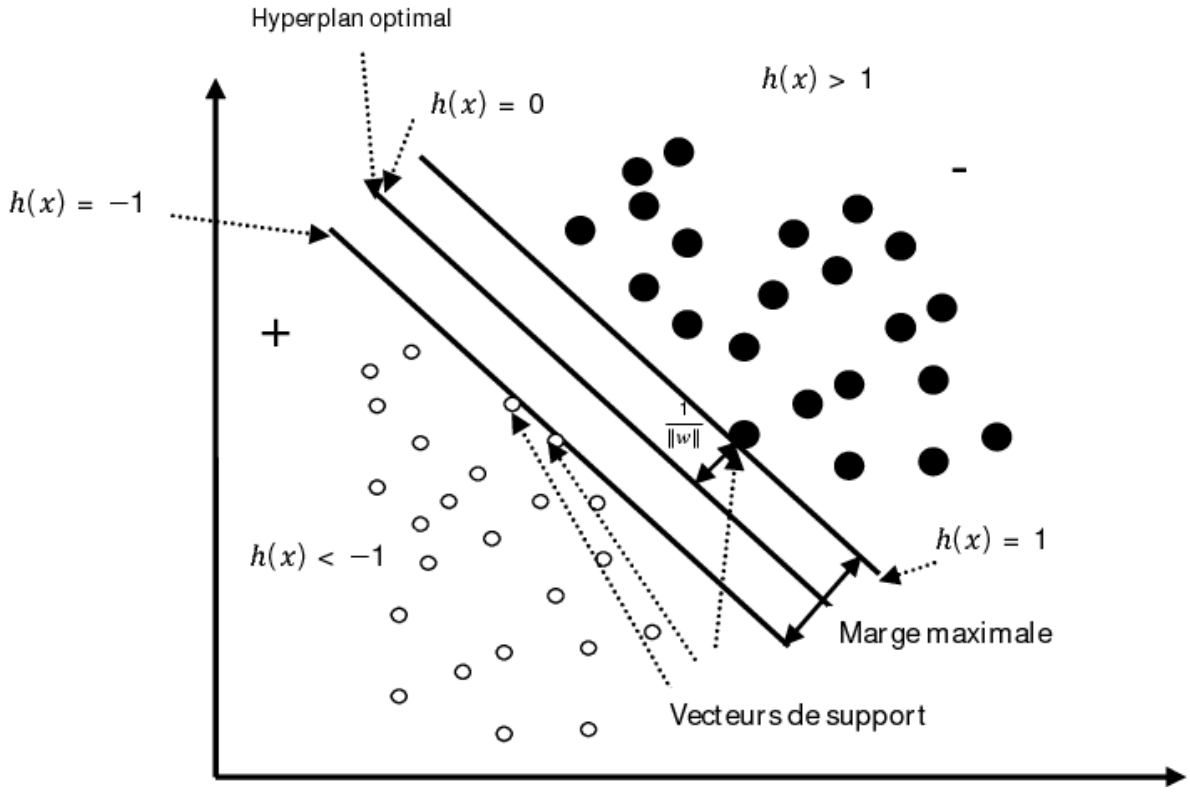

FIGURE 2.3 – Exemple de séparation linéaire dans le plan.

La distance d'un point quelconque x à H est définie par :

$$d_{x,H} = \frac{|w'x + b|}{||w||}.$$

La distance entre H et chacun des deux points x^+ et x^- est donc $\frac{1}{||w||}$. En conséquence, la marge est $\pm\frac{2}{||w||}$. Ainsi, maximiser la marge revient à minimiser $||w||$ sous la contrainte $y_i(w'x_i+b) \geq 1$. Cette contrainte signifie que le SVM prend en compte à la fois la position des exemples par rapport à l'hyperplan, ainsi que leurs distances par rapport à cet hyperplan.
La recherche de l'hyperplan optimale H d'équation $w'x + b = 0$ revient donc à trouver ses paramètres(w, b), maximisant la marge ou minimisant le carée de $||w||^2$ sous des contraintes. Il s'agit là d'un problème d'optimisation d'une fonction objectif sous contraintes :

$$\begin{aligned} &\min_{w} \quad \tfrac{1}{2}||w||^2, \\ &\text{S.C} \quad y_i[w'x_i + b] \geq 1, \quad i \in I = \{1, 2, ..., n\} \end{aligned} \tag{2.5}$$

Il est plus pratique de minimiser $||w||^2$ plutôt que $||w||$.
Il est remarquable que même en supprimant toutes les données qui satisfont l'inigalité de la contrainte, nous pouvons obtenir le même hyperplan. Les données qui vérifient l'égalité de la contrainte sont appelées vecteurs de support, car ce sont les seules données qui contribuent à la détermination de l'hyperplan.
Dans la figure 2.3, les données situées sur les deux droites $+1$ et -1 resprésentent les vecteurs de support.

Au lieu de travailler avec la formulation primale, la plupart des implémentations utilisent la formulation duale de SVM. Le problème 2.5 est transformé en un problème dual équivalent à maximiser, qui introduit les multiplicateurs de lagrange :

$$L(w,b) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i [y_i(w'x_i + b) - 1] \tag{2.6}$$

où les α_i sont les coeficients de lagrange qui doivent être positifs ou non nuls. Le problème 2.5doit satisfaire les conditions de KKT (Karush-Kuhn-Tucker), qui consistent à annuler les dérivées partielles du lagrangian 2.6 par rapport aux variables primales w et b :

$$\begin{cases} \frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0 \\ \\ \frac{\partial L}{\partial w} = 0 \Rightarrow w \sum_{i=1}^{n} \alpha_i y_i x_i \end{cases} \tag{2.7}$$

Le problème $\frac{1}{2}||w||^2$ étant convexe, $w(\alpha)$ est unique. En réinjectant les valeurs obtenues par les conditions KKT dans l'expression 2.6, nous obtenons la forme duale équivalente du problème 2.5 comme suit :

$$\begin{aligned} \min_{\alpha} \quad & L(\alpha) = \tfrac{1}{2} \textstyle\sum_{j=1}^{n} \sum_{j=1}^{n} (y_i y_j x_i' x_j) \alpha_i \alpha_j - \sum_{j=1}^{n} \alpha_i \\ \text{S.}c \quad & \textstyle\sum_{j=1}^{n} y_i \alpha_i = 0 \\ & 0 \leq \alpha_i, i \in I. \end{aligned} \tag{2.8}$$

C'est un problème quadratique de dimension n (taille de l'ensemble d'apprentissage).
Sa résolution permet de trouver les indices i des multiplicateurs optimaux $\hat{\alpha_i}$ positifs correspondant aux point $\hat{x_i}$ qui sont les vecteurs de support, et leur ensemble d'indices correspondant est :

$$I_{sv} = i \in I : \hat{\alpha_i} > 0$$

Une fois la solution optimale $\hat{\alpha}$ du problème dual obtenue, le vecteur de poids w de l'hyperplan de marge maximale souhaitée est :

$$\hat{w} = \sum_{j=1}^{n} \hat{\alpha} y_i x_i = \sum_{i \in I_{sv}} \hat{\alpha} y_i x_i \tag{2.9}$$

La valeur optimal $\hat{b}$ est calculée en considérant une moyenne de l'enssemble des vecteurs de support, Alors nous obtenons la valeur :

$$\hat{b} = \frac{1}{|I_{sv}|} \sum_{i \in I_{sv}} [y_i - \hat{w}' x_i]. \tag{2.10}$$

La régle de classification d'une nouvelle observation x, sur la basse d'un hyperplan de marge maximale est donnée par :

$$h(x) = sign\left(\sum_{i \in I_{sv}} (\hat{\alpha_i} y_i x_i' x) + \hat{b} \right) \tag{2.11}$$

2.4.1.2 SVM a marge souple

Souvent il arrive que même si le problème est linéaire, les données sont affectées par un bruit et les deux classes se retrouvent mélangées autour de l'hyperplan de séparation. Pour gérer ce problème, on modifie les SVM pour maximiser la marge tout en tolérant quelques erreurs de classification. Cela se fait avec le paramètre C, qui équilibre la régularisation et l'ajustement aux données. Trouver la meilleure valeur pour C est difficile et nécessite souvent une validation croiséelors des simulations pratiques[37].

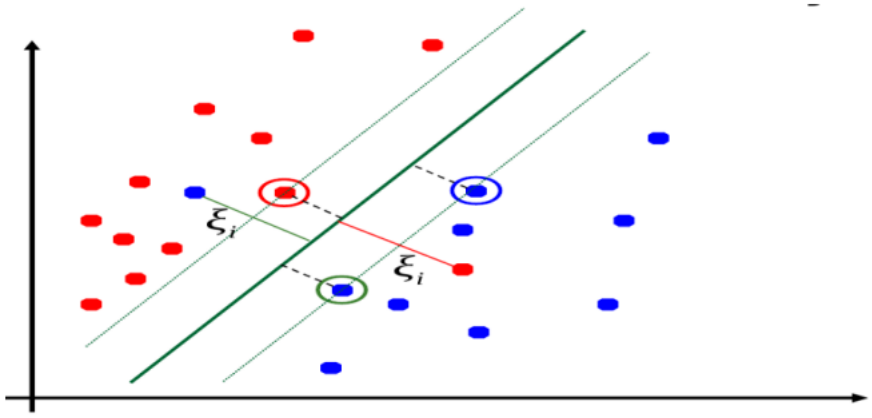

FIGURE 2.4 – Marge souple et variable d'ecart ξ_i

Le programme CSVM (ou SVM à marge souple) s'écrit donc en associant à chaque exemple (x_i, y_i) une variable d'écart ξ_i, qui mesure à quel point l'exemple est mal classifié. L'erreur empirique s'exprime sous la forme de la somme des variables d'écart sur l'échantillion.
Pour un hyperpian (w, b) et un exemple (x_i, y_i) la pénalisation ξ_i associée sera :

$$\xi_i = \begin{cases} 0, & \text{si} \quad y_i(w^{'} x_i + b) \geq 1 \\ \\ 1 - y_i(w^{'} x_i + b), & \text{sinon.} \end{cases} \tag{2.12}$$

Le programme primal d'un CSVM à marge souple s'écrit alors :

$$\begin{aligned} \min_{x,\xi} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i \\ \text{S.c} \quad & y_i[w^{'} x_i + b] \geq 1 - \xi_i, \\ & \xi_i \geq 0, i \in I = \{1, 2, ..., n\} \end{aligned} \tag{2.13}$$

Le paramètre de régularisation C contrôle la misclassification de l'échantillon, et il est utilisé pour pénaliser les variables ξ_i. Une valeur élevée de C correspond à une pénalité importante des erreurs.

Comune pour le problème à marge dure, on introduit la version duale de CSVM :

$$\begin{gathered}\min_{\alpha} L(\alpha) = \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n}(x_i y_j x_i^{'} x_j)\alpha_i\alpha_j - \sum_{i=1}^{n}\alpha_i \\ \textstyle\sum_{i=1}^{n} y_j\alpha_i = 0 \\ 0 \leq \alpha_i \leq C, i \in I\end{gathered} \tag{2.14}$$

La fonction de Lagrange associée à ce problème dual est la suivante :

$$G(\alpha, b, s, \xi) = \frac{1}{2}\alpha^{'} D\alpha - e^{'}\alpha + by^{'}\alpha - s^{'}\alpha + \xi^{'}(\alpha - Ce), \tag{2.15}$$

où $y = y(I) = (y_i, i \in I)$, la matrice D, définie par ces éléments $d_{i,j} = y_i y_j x_i^{'} x_j$, est carrée d'ordre n, symétrique et semi définie positive, et le n-vecteur non-négatif s représente des variables d'écart.
Les conditions d'optimalité de KKT des deux problèmes pricédents s'écrivent comme suit :

$$\begin{array}{l}\frac{\partial G}{\partial \alpha} = D\alpha - e + by - s + \xi = 0, \\ \frac{\partial G}{\partial b} = y'\alpha = 0, \\ \xi_i(\alpha_i - C) = 0, \text{si} \quad \alpha_i = 0, \quad i \in I \\ s_i \geq 0, \quad \xi_i \geq 0, \quad i \in I \\ 0 \leq \alpha_i \leq C, \quad i \in I\end{array} \tag{2.16}$$

Les conditions de cromplémentarité de KKT sont très utiles pour les problèmes de SVM, car elles permettent d'en extraire les éléments representatifs de l'échantillon d'apprentissage. En fonction de la valeur optimale des multiplicateurs de Lagrange α_i, les exemples d'apprentissage correspondants x_i peuvent être classés en trois catégories :

— $\alpha_i = 0 \Rightarrow \xi_i = 0$: L'exemple x_i est bien classé, mais il n'est pas un vecteur support et il ne sera pas retenu pour construire le modèle final.

— $0 \leq \alpha_i \leq C \Rightarrow y_i(w^{'}x_i + b) - 1 + \xi_i = 0$ et $\xi_i = 0 \Rightarrow y_i(w^{'}x_i + b) = 1$: l'exemple correspondant x_i, est bien classé et situé sur la marge, donc c'est un vecteur de support il sera utilisé pour construire le classificateur final.

— $\alpha_i = C \Rightarrow y_i(w'x_i + b) - 1 + \xi_i = 0$ et $\xi_i \leq 0$: puisque α_i est différent de zéro, alors l'exemple x_i est considéré comme un vecteur de support borné. Si $0 \leq \xi_i \leq 1, x_i$ est bien classé, et si $\xi_i \geq 1, x_i$ est mal classé.

On déduit que les seuls coefficients α_i non nuls sont ceux associés aux exemples x_i de l'ensemble d'apprentissage S qui sont sur la marge ($i.e., \alpha_i = C$ et $0 \leq \xi_i \leq 1$) ou à l'intérieur de l'espace délimité par la marge ($i.e., 0 \leq \alpha_i \leq C$ et $\xi_i = 0$). Ces exemples sont les vecteurs de support et leur ensemble d'indices correspondant est :

$$I_{sv} = \{i \in I : (\alpha_i = C \quad \text{et} \quad 0 \leq \xi_i \leq 1) \vee (0 < \alpha_i < C \quad \xi_i = 0)\}$$

Le calcul des paramètres optimaux $\hat{w}$ et $\hat{b}$ de l'hyperplan séparateur H, ainsi que la fonction de classification se font de la même façon que le cas précèdent.

2.4.2 SVM non linéaire

Reconnaître la mauvaise classification de certains exemples ne conduit pas toujours à une généralisation efficace pour un hyperplan, même lorsqu'il est optimisé. Au lieu d'une ligne droite, la représentation idéale de la fonction de décision serait celle qui adhère étroitement aux données d'entraînement.

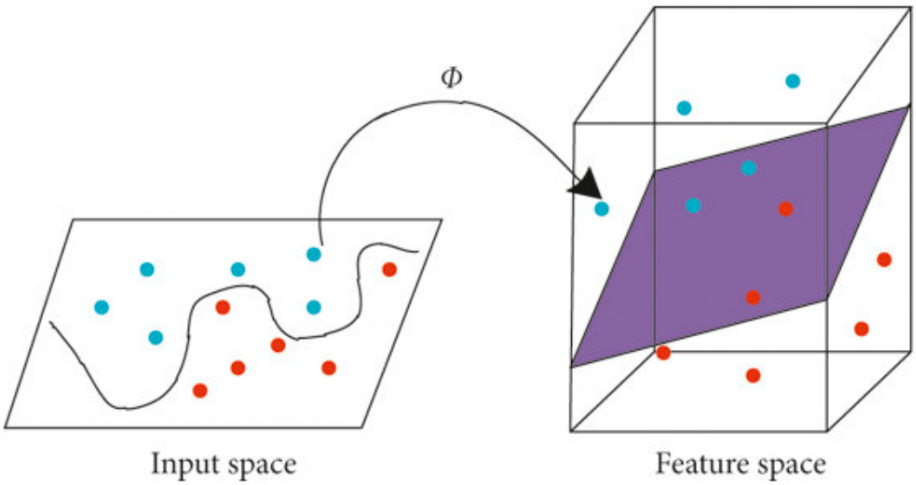

FIGURE 2.5 – Passage des données de l'espace d'entrée vers un espace de description où les données sont linéairement séparables.

Lorsqu'il est difficile voire impossible de déterminer une fonction non linéaire dans l'espace d'origine, les données sont souvent transformées dans un nouvel espace appelé espace de caractéristiques (Features space), où cette fonction devient linéaire (voir la figure 2.5). Cette transformation est réalisée à l'aide d'une fonction de mappage $\phi(x)$, et la fonction objectif du problème d'optimisation est réécrite en fonction des produits scalaires dans ce nouvel espace. Dans ce nouvel espace, la fonction objectif devient :

$$L(\alpha) = \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} y_i y_j < \phi(x_i), \phi(x_j) > \alpha_i \alpha_j - \sum_{i=1}^{n} \alpha_i \tag{2.17}$$

où $< \phi(x_i), \phi(x_j) >$ est le produit scalaire des deux images des vecteurs x_i et x_j dans le nouvel espace et dont le résultat est un scalaire.
Pour calculer l'optimum de la fonction 2.17, on utilise une astuce appelée $Noyau(Kernel)$, au lieu de calculer directement le produit scalaire $< \phi(x_i), \phi(x_j) >$, on calcule une fonction $K(x_i, x_j)$ qui représente ce produit scalaire dans le nouvel espace, sans avoir besoin de connaître la transformation ϕ. La fonction $K(x_i, x_j)$ peut être vue comme une matrice de Gram $G \in R^{n \times n}$ [15] représentant les distances entre tous les exemples :

$$\begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix}$$

Pour qu'une matrice G (fonction K) soit un noyau, elle doit respecter les conditions de Mercer [20], c'est-à-dire être semi-définie positive (symétrique et ne possédant pas de valeurs propres

négatives). Certains noyaux couramment utilisés et qui sont considérés comme standards. Une fois le noyau choisi, la fonction objectif 2.17 peut être calculée comme suit :

$$L(\alpha) = \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} y_i y_j K(x_i, x_j)\alpha_i\alpha_j - \sum_{i=1}^{n}\alpha_i. \tag{2.18}$$

Une fois que la solution optimale duale $\hat{\alpha}$ est obtenue, les paramètres optimaux $\hat{w}$ et $\hat{b}$ de l'hyperplan séparateure H se font de la même façon que le cas précèdent. Et la fonction de décision devient :

$$f(x) = sign\left(\sum_{i\in I_{sv}} (\hat{\alpha}_i y_i k(x, x_i) + \hat{b}\right) \tag{2.19}$$

2.5 Régularisation par les SVM

La régularisation L1 et L2 sont des techniques couramment utilisées en apprentissage automatique pour prévenir le surapprentissage et améliorer la capacité de généralisation des modèles. En SVM, la régularisation est essentielle pour contrôler la complexité du modèle. La régularisation L1, également appelée Lasso, favorise la parcimonie et la sélection de caractéristiques en forçant certains coefficients à zéro. En revanche, la régularisation L2, connue sous le nom de Ridge, pénalise les grands poids sans les forcer à zéro. Ces deux approches complémentaires jouent un rôle crucial dans l'optimisation des SVMs pour des performances optimales en termes d'ajustement et de généralisation des modèles.

Nous supposons que les données sont des couples $(x_i, y_i) \in X \times Y, i \in I = \{1, 2, ...n\}$. X désigne l'espace des variables explicatives (espace d'entrée) souvent pris dans $\mathbb{R}^p$, $y_i \in -Y$={-1,1} est l'ensemble des classes (espace de sortie) et n est la taille de l'échantillon.
On cherche un hyperplan (H) d'équation :

$$w'x + b = 0$$

On pose $W = \begin{pmatrix} w \\ b \end{pmatrix}$, $X_i' = (x_i', 1)$
Soit la fonction objectif d'un problème d'optimisation régularisée :

$$\min_{w,b} \quad \frac{1}{2}w^T w + \frac{1}{2}b^2 + C\sum_{i=1}^{n}\xi(w, x_i, y_i) \tag{2.20}$$

2.5.1 L2-régularisé avec perte L1 :

Perte L1 : La perte L1, ou erreur absolue moyenne, est définie comme la somme des valeurs absolues des erreurs. Mathématiquement, pour des prédictions $\hat{y}_i$ et des vraies valeurs y_i, elle est donnée par :

$$\text{L1} = \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Le problème primal des SVMs est formulé comme suit :

$$\min_{w} \quad \frac{1}{2}w^T w + c\sum_{i=1}^{n}(max(0, 1 - y_i w' x_i)) \tag{2.21}$$

Le problème dual des SVM est formulé comme suit :

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T \bar{Q}\alpha - e^T\alpha \\ & 0 \leq \alpha_i \leq C \end{aligned} \tag{2.22}$$

e est un n-vecteur formé de 1 et $\bar{Q} = Q + D$ oú D est une matrice diagonale et on a :
$Q_{i,j} = y_i y_j x_i' x_j$
$D_{ii} = 0, \quad \forall i$

2.5.2 L2-regularisé avec perte L2 :

Perte L2 : La perte L2, ou erreur quadratique moyenne, est définie comme la somme des carrés des erreurs. Mathématiquement, pour des prédictions $\hat{y}_i$ et des vraies valeurs y_i, elle est donnée par :

$$\text{L2} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Le problème primal des SVM est formulé comme suit :

$$\min_{w} \quad \frac{1}{2}w^T w + c\sum(\max(0, 1 - y_i w' x_i))^2 \tag{2.23}$$

Le problème dual des SVM est formulé comme suit :

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T \bar{Q}\alpha - e^T\alpha \\ & \alpha_i \geq 0, \quad \forall i \end{aligned} \tag{2.24}$$

$\bar{Q} = Q + D$, avec $D_{ii} = \frac{1}{2C}, \quad \forall i$

2.6 Méthode de résolution

Afin de trouver les paramètres de l'hyperplan séparateur, il nécessaire de résoudre un programme quadratique, dont la formulation daule pour un SVM non linéaire est la suivante :

$$\begin{aligned} & min_{\alpha} \frac{1}{2}\alpha^{'} D\alpha + e^{'}\alpha, \\ & y^{'}\alpha = 0, \\ & 0 \leq \alpha \leq Ce, \end{aligned} \tag{2.25}$$

où α et y sont des n vecteurs, et e est également un n vecteur dont chaque composante est égale à 1. La matrice de Gram-Schmidt $D_{ij} = y_i y_j k(x_i, x_j)$ est une matrice carrée d'ordre n, symétrique, et semi-définie positive lorsque le noyau k vérifie les conditions de Mercer[20].
Toutes les techniques et logiciels développés pour la programmation quadratique convexe peuvent résoudre ce type de problème. Si la matrice D est dense, alors les méthodes d'activation de contraintes résolvent efficacement le problème (2.25). Cependant, si la taille de l'échantillon n est très grande et que D est cereuse, les méthodes des points intérieurs constituent un meilleur

choix. Malheureusement, pour les problèmes SVM rencontrés en pratique, la matrice D est à la fois dense et de grande taille, ce qui pose un défi pour les deux approches.

Pour le problème SVM (2.25), à l'optimum $\hat{\alpha}$, une grande partie des variables duales α sont soit nulles, soit fixées à C. C'est cette caractéristique qu'exploitent la plupart des méthodes conçues pour les SVM, et par conséquent, elles résolvent un sous-problème quadratique (SQP) de petite taille à chaque itération.

2.6.1 Les méthodes de décomposition

Ces méthodes reposent sur le concept de décomposition d'un problème à grande échelle en une série de sous-problèmes plus petits et plus faciles à résoudre. L'optimisation est effectuée de manière itérative sur une partition de l'ensemble de données d'entraînement en résolvant des systèmes linéaires de faible dimension.

Méthodes de Chunking

Cette méthode, initialement décrite par Vapnik [3], fonctionne en éliminant progressivement les points inactifs ($\alpha_i = 0$) au fur et à mesure de la progression de l'algorithme. L'idée est de résoudre le problème quadratique (PQ) pour un sous-ensemble de points à chaque étape. Le sous-ensemble suivant se compose des vecteurs de support de la solution précédente et des points m sélectionnés dans l'échantillon restant, choisis pour leur violation maximale des conditions de KKT. Ainsi, de sous-groupe en sous-groupe, tout le problème est résolu. La limite de cette méthode réside dans la taille de la solution finale, car il peut ne pas être possible de stocker l'intégralité de la matrice du noyau en mémoire. Néanmoins, cette méthode fonctionne assez bien pour des problèmes impliquant quelques centaines de vecteurs supports.

Méthodes de décomposition séquentielle

Cette méthode, proposée par Osuna [2], stipule qu'une succession de sous-problèmes quadratiques, ayant au moins un exemple qui viole les conditions KKT, converge toujours vers une solution optimale. Elle reprend le principe de décomposition du Chunking, mais sur des sous-ensembles de mille fixe arequels il est possible d'ajouter ou de retirer un exemple à chaque fois. En pratique, pour accélérer la convergence, on utilise des heuristiques permettant d'inclure ou d'exclure quelques exemples de la fonction objectif. Ceci permet d'entrainer de grands ensembles de données. Des tech- niques de caching sont aussi utilisées pour traiter des problèmes de quelques milliers de vecteurs supports. Ces techniques permettent une meilleure gestion de l'espace mémoire en ne sauvegardant que les valeurs fréquemment utilisées (les autres sont effacées).

Méthode de minimisation séquentielle

La méthode d'optimisation minimale séquentielle (SMO), introduite par Platt [4], peut être considérée comme un cas extrême de méthodes de décomposition successives. A chaque itération, il résout un problème quadratique avec seulement deux variables, et la solution est obtenue analytiquement sans utiliser de solveur. Le choix de la paire de variables (α_i, α_j) à optimiser est déterminé par des heuristiques. Une variante proposée par l'auteur consiste à les sélectionner de telle sorte qu'ils modifient la fonction objectif dans des sens opposés.

Méthodes d'activation des contraintes

Pour résoudre efficacement les problèmes SVM, diverses méthodes ont vu le jour, notamment l'approche de Scheinberg [5], Fan et al. et autres. Ces méthodes emploient également le principe de décomposition en fixant une partie significative des variables à zéro ($\alpha_i = 0$) et en résolvant un problème quadratique réduit (PQ) sur les variables restantes. Différentes implémentations basées sur cette approche peuvent être distinguées, telles que SVM-QP et SVM-RSQP [16].

2.7 Les domaines d'applications des SVM :

SVM est une méthode de classification qui montre de bonnesperformances dans la résolution des problèmes variés. Cette méthode a montré son efficacité dans de nombreux domaines d'applications tels que :

1. Classification de données biologiques/physiques.
2. Classification de documents numériques.
3. Classification d'expressions faciales.
4. Classification de textures.
5. E-learning.
6. Détection d'intrusion.
7. Reconnaissance de la parole.
8. CBIR : Content Based Image Retrieval.

2.8 Les avantages et les inconvénients des SVM :

Avantages :

- Un grand taux de classification et de généralisation par rapport aux méthodes classiques.
- Elle nécessite moins d'effort pour designer l'architecture adéquate (petit nombre de paramètres à régler ou à estimer).
- La résolution du problème est convertie en résolution d'un problème quadratique convexe dont la solution est unique et donnée par des méthodes mathématiques classiques de programmation quadratique.

Inconvénients :

- L'inconvénient majeur du classificateur SVM est qu'il est désigné ou conçu pour la classification binaire (la séparation entre deux classes une +1 et l'autre -1).

2.9 Conclusion

Dans ce chapitre, nous avons tenté de présenter de manière simple et complète le concept de système d'apprentissage introduit par Vladimir Vapnik, les Machines à Vecteurs de Support. Nous avons donné une vision générale et une vision purement mathématique des SVM. Cette méthode de classification est basée sur la recherche d'un hyperplan qui permet de séparer au mieux des ensembles de données. Nous avons exposé le cas linéairement séparable et les cas non linéairement séparables qui nécessitent l'utilisation de fonction noyau (kernel) pour changer d'espace. Cette méthode est applicable pour des tâches de classification à deux classes.

Chapitre 3

Méthode support pour la programmation quadratique convexe à variables bornées

La méthode directe du support (MDS) de Gabassov dans les années 70[1], permet de résoudre des problèmes de programmation quadratique convexes. Elle constitue une généralisation de l'algorithme du simplexe en utilisant la métrique de ce dernier, ce qui permet de modifier un seul indice non basique à chaque itération. Une itération de la MDS consiste à trouver une direction d'amélioration et un pas optimal tout au long de cette direction afin d'optimiser l'objectif [31]. Elle permet ainsi de générer une suite finie de points réalisables convergeant vers la solution optimale du problème.
Dans ce chapitre, on expose l'algorithme de la MDS pour la résolution d'un PQC à variables bornées , puis nous l'appliquant sur un exemple numérique .

3.1 Postion du problème et définition

Un problème de programmation quadratique à variable bornées se presente sous la forme suivante :

$$\begin{cases} \min_x F(x) = & \frac{1}{2}x'Dx + c'x \\ & l \leq x \leq u, \end{cases} \tag{3.1}$$

où $D' = D \geq 0$ est une matrice carrée d'ordre n, symétrique et supposée semi-définie positive. x est un n-vecteur des variables de décision, l, u et c sont des n-vecteur, avec $l_j \leq u_j, \quad \forall j \in J$. Notons par $J = \{1, 2, \ldots, n\}$ l'ensemble des indices des variables de décision x et représentant aussi les lignes et les colonnes de la matrice hessienne D.
J_B : ensemble des indices des variables basiques.
$J_N = J|J_B$: est un enssemble d'indices des variables non basique.
x_B est une variable basique.

3.2 Conditions d'optimalité de KKT

3.2.1 Fonction de Lagrange :

La fonction de Lagrange associée au problème 3.1 est la suivante :

$$F(x,s,\lambda) = \frac{1}{2}x^{'}Dx - c^{'}x - s^{'}(x-l) + \lambda'(x-u), \tag{3.2}$$

Les n-vecteurs s et λ sont respectivement les multiplicateurs de Lagrange associés aux contraintes de bornes du problème (3.1) , tel que :

— s est associé aux contraintes d'inégalité des bornes inférieure, telles que $(l \leq x)$.

— λ est associé aux contraintes des bornes supérieures, telles que $(x \leq u)$.

Les conditions d'optimalité de KKT [14] pour ce problème seront obtenues en dérivant le lagrangien par rapport à x.

$$\frac{\partial L}{\partial x} = Dx + c + s - \lambda = 0,$$
$$\lambda_i(x_i - u) = 0, \quad s_i(x_i - l) = 0, \quad i = 1 \ldots n,$$
$$s_i \geq 0, \quad \lambda_i \geq 0, \quad i = 1 \ldots n,$$
$$l \leq x \leq u, \quad i = 1 \ldots n,$$

3.3 Définitions et concepts de base

Définition 3.1. (Solution réalisable) : On appelle solution réalisable (SR) du problème (PQC), tout vecteur $x \in \mathbb{R}^n$ vérifiant la relation 3.1

Définition 3.2. (solution optimale) : On appelle solution optimale du problème (PQC), tout point x^* qui vérifie :

$$F(x^*) = \frac{1}{2}(x^*)'Dx^* + c'x^* = \min_x \left(\frac{1}{2}x'Dx + c'x\right)$$

Définition 3.3. (Solution ϵ-optimale) : On appelle solution *ϵ-optimale* ou *suboptimal* du problème (PQC), toute solution réalisable x^ϵ vérifiant :

$$F(x_\varepsilon) - F(x^*) = (\frac{1}{2}x'_\varepsilon Dx_\varepsilon + c'x_\varepsilon) - (\frac{1}{2}x'^* Dx^* + c'x^*) \leq \varepsilon$$

où ε est un nombre arbitraire positif ou nul, choisi comme une précision de calcul.

Définition 3.4. (Vecteur des estimations) : Le gradient de la fonction F au point x est représenté par :

$$E = \nabla F(x) = Dx + c.$$

Ce vecteur E est également appelé *vecteur des estimations* ou *vecteur des coûts réduits.*

Définition 3.5. (Support de la fonction objectif) : Un ensemble d'indices $J_B \subset J$ est appelé *support du problème* (3.1) si la sous-matrice associée $D_B = D(J_B, J_B)$ est non singulière.

Définition 3.6. (Solution réalisable de support) : Le couple $\{x, J_B\}$ formé de la solution x et du support J_B, est dit *solution réalisable de support (SRS)* si x est une solution réalisable pour (3.1) et $E_B = 0$, $\forall j \in J_B$. Elle est dite non dégénérée si

$$l_j < x_j < u_j, \quad \forall j \in J_B. \tag{3.3}$$

3.4 Formule d'accroissement de la fonction objectif

Soit $\{x, J_B\}$ une SRS des contraintes du probléme (PQC) et considérons une autre SRS quelconque $\bar{x} = x + \nabla x$. L'accroissement de la fonction objectif s'écrit comme suit[22, 27] :

$$\begin{aligned} F(\bar{x}) - F(x) &= \frac{1}{2}\bar{x}'D\bar{x} + c'\bar{x} - \frac{1}{2}x^T Dx - c'x \\ &= \frac{1}{2}(x+\Delta x)'D(x+\Delta x) + c'(x+\Delta x) - \frac{1}{2}x'Dx - c'x \\ &= \frac{1}{2}(\Delta x)'D\Delta x + (\Delta x)'(Dx+c). \end{aligned}$$

L'accroissement est donc :

$$F(\bar{x}) - F(x) = E'\Delta x + \frac{1}{2}\Delta x'D\Delta x \tag{3.4}$$

où $E = Dx + c$ est le gradient de la fonction objectif F au point x, tel que : $E = (E_j, j \in J)$, et se décompose en composante basique et non basique :

$$E = \begin{pmatrix} E_B \\ E_N \end{pmatrix}, \text{ avec } E_B = (E_j \ j \in J_B) \text{ et } E_N = (E_j \ j \in J_N).$$

3.5 Critères d'optimalité

Théorème 3.1. *[14] Soit $\{x, J_B\}$ une SRS du problème* (3.1)*. Alors, les relations suivantes :*

$$\begin{cases} E_j \geq 0, & si \quad x_j = l_j \\ E_j \leq 0, & si \quad x_j = u_j \\ E_j = 0, & si \quad l_j \leq x_j \leq u_j \end{cases} \tag{3.5}$$

sont suffisantes pour l'optimalité du point x, et sont aussi nécessaires dans le cas où la SRS est non dégénérée[27].

3.6 Crétère de sub-optimalité

La quantité $\beta(x, J_B)$ suivante, définie par :

$$\beta(x, J_B) = \sum_{J \in J_{NN}, E_j > 0} E_j(x_j - l_j) + \sum_{j \in J_{NN}, E_j < 0} E_j(x_j - u_j) \tag{3.6}$$

est appelée estimation de sub-optimalité de la solution réalisable de support $\{x, J_B\}$. Elle mesure l'écart entre la valeur de la fonction objectif à une solution donnée x et celle d'une solution optimale x^0, i.e, $\beta(x, J_B) = f(x) - f(x^*)$.

Le critère de sub-optimalité est décrit dans le théorème suivant :

Théorème 3.2. ***(condition suffisante de suboptimalité)***
Soit $\{x, J_B\}$ une SRS du problème (3.1) *et ϵ un nombre réel positif ou nul. Si $\beta(x, J_B) \leq \epsilon$, alors x est une solution ϵ-optimale (sub-optimale) du problème* (3.1)*.*

Le théorème précedent décrit la condition suffisante pour qu'un point x soit ϵ-optimale. De plus, pour $\epsilon = 0$, il donne la condition suffisante pour l'optimilalité d'une SRS $\{x, J_B\}$.

3.7 Itération

Soit $\{x, J_B\}$ une solution réalisable de support accordée telle que $\beta(x, J_B) > 0$. Nous allons décrire en détail l'itération $\{x, J_B\} \rightarrow \{\bar{x}, \bar{J}_B\}$. La première partie consiste à transformer la solution réalisable x en une nouvelle SR $\bar{x} = x + \theta d$, où d est un vecteur représentant la direction de decente et θ un nombre réel positif ou nul, appelé le pas le long de d, la deuxième étape sera celle du changement du support J_B en $\bar{J}_B$.
Pour le choix de la direction d on doit tenir compte des points suivants :

a) La relation $\bar{E}_j = 0$, $j \in J_B$, doit etre vérifiée pour la nouvelle SRS $\bar{x} = x + \theta d$.

b) La fonction objectif doit diminuer en passent de x a $\bar{x}$, en utilisant les indices $j \in J_N$.

A l'itération courante, soit J_{NN0} l'ensemble d'indice des variables non basiques ne vérifiant pas le critére d'optimalité (3.5) et qui est défini comme suit :

$$J_{NN0} = \Big\{ j \in J_N : [x_j < u_j, E_j < 0] or [x_j > l_j, E_j > 0] \Big\} \tag{3.7}$$

De la relation $\bar{E}_j = E_j(x + \theta d) = 0$, $\forall j \in J_B$, alors il s'ensuit que d doit vérifier $t_j = (Dd)_j = 0$, $\forall j \in J_B$, c'est à dire que les composantes basiques t_B du n-vecteur t vérifient la relation suivante :

$$D(J_B, J_B)d(J_B) + D(J_B, J_N)d(J_N) = 0$$

d'où

$$d(J_B) = -D_B^{-1} D(J_B, J_N) d(J_N) \tag{3.8}$$

Par conséquent, quelque soit le choix de la composante $d(J_N)$ du vecteur d, la relation $E_j(x + \theta d) = 0$, $j \in J_B$, reste toujours vérifiée.

Soit $j_0 \in J_{NNO}$ un indice non basique pour lequel le critère d'optimalité (3.5) n'est pas vérifié. La direction de descente $d = (d_B, d_N)'$, avec $d_B = (d_j, j \in J_B)$ et $d_N = (d_j, j \in J_N)$, est calculée comme suit :

$$d_j = \begin{cases} -\text{sign}\,(E_{j_0})\,, & \text{si} \quad j = j_0 \\ 0, & \text{sinon} \end{cases} \tag{3.9}$$

En utilisant (3.8), la relation (3.7) devient :

$$d(J_B) = D_B^{-1} D(J_B, j_0) sign E_{j_0}. \tag{3.10}$$

Le choix du pas optimal θ le long de la direction d se fait de telle sorte que :

(i) Les contraintes du PQ (3.1) doivent être vérifiées pour $\bar{x} = x + \theta d$.

(ii) Le passage de x à $\bar{x}$ doit assurer une diminution maximale de valeur de la fonction objectif.

Désignons par θ_N le pas maximal sur les composantes non basiques, tels que :

$$l_j - x_j \leq \theta d_j \leq u_j - x_j, \quad \forall j \in J_N \tag{3.11}$$

et par θ_{j_1} le pas maximal sur les composantes basiques, tels que :

$$l_j - x_j \leq \theta d_j \leq u_j - x_j, \quad j \in J_B \tag{3.12}$$

La condition (3.10)est toujours remplie pour tout θ si $j \neq j_0$.

Pour $j = j_0$ le pas θ_{j_0} doit vérifier :

$$\theta_{j_0} \leq (l_{j_0} - x_{j_0}) \leq d_{j_0} \quad \text{si} \quad d_{j_0} < 0$$

$$\theta_{j_0} \leq (u_{j_0} - x_{j_0}) \leq d_{j_0} \quad \text{si} \quad d_{j_0} > 0$$

En tenant compte de (3.8) on aura donc :

$$\theta_N = \theta_{j_0} = \begin{cases} u_{j_0} - x_{j_0} & \textbf{si} \quad E_{j_0} < 0 \\ x_{j_0} - l_{j_0} & \textbf{si} \quad E_{j_0} > 0 \end{cases} \tag{3.13}$$

On procédant de la même manière, on aura pour la condition (3.11) le pas maximal suivant :

$$\theta_B = \theta_{j_1} = min\ \theta_j, j \in J_B$$

Où :

$$\theta_j = \begin{cases} \frac{u_j - x_j}{d_j} & \textbf{si} \quad d_j > 0 \\ \frac{l_j - x_j}{d_j} & \textbf{si} \quad d_j < 0 \\ +\infty & \textbf{si} \quad d_j = 0. \end{cases} \tag{3.14}$$

Ensuite, déterminons le pas θ_F pour lequel la condition ii) est satisfaite. Ceci est traduit par l'équation suivante :

$$\frac{dF(x+\theta d)}{d\theta} = \frac{\partial F(x+\theta d)}{\partial x} \cdot \frac{\partial (x+\theta d)}{\partial \theta} = 0.$$

Ce qui nous donne :

$$(Dx + c)'d + \theta d' D d = 0$$

On prendra donc θ_F tel que :

$$\theta_F = \begin{cases} \frac{-E_{j_0}}{t_{j_0}}, & \text{si } E_{j_0} t_{j_0} < 0, \\ \infty, & \text{sinon ,} \end{cases} \tag{3.15}$$

Où

$$E_{j_0} t_{j_0} = E' d + \theta d' D d = D(j_0, j_0) - D(j_0, J_B) D_B^{-1} D(J_B, j_0). \tag{3.16}$$

Par conséquent, on choisira θ tel que :

$$\theta = \min\{\theta_{j_0}, \theta_{j_1}, \theta_F\}. \tag{3.17}$$

Changement de support $J_B \rightarrow \bar{J}_B$

Trois cas peuvent alors se présenter :

— **Si** $\theta = \theta_{j_0}$ **:**
On pose $\bar{J}_B = J_B$ et on regarde si $\{\bar{x}, \bar{J}_B\}$ est une SRS optimale du problème (3.1).

— **Si** $\theta = \theta_{j_1},\ j_1 \in J_B$ **:**
Dans ce cas, on aura :

$$\bar{x}_{j_1} = \begin{cases} u_{j_1}, & \text{si } d_{j_1} > 0, \\ l_{j_1}, & \text{si } d_{j_1} < 0, \text{ et } j_1 \in J_B. \end{cases}$$

La composante $\bar{x}_{j_1}$ devient alors critique et on retire l'indice j_1 de J_B en posant

$$\bar{J}_B = J_B \setminus j_1.$$

La matrice $\bar{D}_B = D(\bar{J}_B, \bar{J}_B)$, obtenue à partir de D_B en supprimant la ligne et la colonne d'indice j_1, est inversible car la matrice D_B est définie positive.
Puisque $E_j(\bar{x}) = 0$, $j \in \bar{J}_B$, alors $\bar{J}_B$ est un support accordé de la fonction F. On reprend ainsi le même procédé en partant de $\{\bar{x}, \bar{J}_B\}$.

— **Si :** $\theta = \theta_F$
Dans ce cas, on a $E_{j_0}(\bar{x}) = 0$ et on pose :

$$\bar{J}_B = J_B \cup j_0.$$

La matrice $\bar{D}_B = D(\bar{J}_B, \bar{J}_B)$ est obtenue en ajoutant à la matrice D_B la ligne $D(j_0, \bar{J}_B)$ et la colonne $D(\bar{J}_B, j_0)$. Puisque $\alpha > 0$, alors $\det = D(\bar{J}_B, \bar{J}_B) \neq 0$. De plus, on a $E_j(\bar{x}) = 0$, $j \in \bar{J}_B$, alors $\bar{J}_B$ est un support accordé de la fonction F. Si le critère d'optimalité n'est pas vérifié, on recommence alors une nouvelle itération à partir de $\{\bar{x}, \bar{J}_B\}$.

Algorithm 1 Algorithme de résolution d'un PQC sans contraintes

Début

S1 :Initialisation

Soit $\{x, J_B\}$ une SRS accordée du problème (3.1), ϵ est un nombre arbitraire positif ou nul, avec $E = Dx + c$;

S2 : Test d'optimalité et de sub-optimalité de $\{x, J_B\}$

Calculer $\beta(x, J_B)$ avec la formule (3.6) ;

Si $\beta(x, J_B) = 0$, **Alors**

$\{x, J_B\}$ est optimal, alors aller à fin .

Fin Si

Si $\beta(x, J_B) \leq \epsilon$, **Alors**

$\{x, J_B\}$ est ϵ-optimal, alors aller à fin.

Sinon aller à l'etape **S3**

Determiner l'ensemble d'indice des variables non basiques

$$J_{NN0} = \Big\{ j \in J_N : [x_j < u_j, E_j < 0] or [x_j > l_j, E_j > 0] \Big\}$$

Choisir un indice j_0 à partir de J_{NNO} tel que :

$$|E_{j_0}| = \max_{j \in J_{NNO}} |E_j| \tag{3.18}$$

S3 : Changement de la SR

- **Calculer** $d = (d_B, d_N)'$ en utilisant les relation (3.9) et (3.10).
- **Déterminer** le pas optimal par :

  $$\theta = \min\{\theta_{j_0}, \theta_{j_1}, \theta_F\},$$

  où θ_{j_0}, θ_{j_1} et θ_F sont respectivement déterminées par (3.13), (3.14) et (3.15).
- **Calculer** la direction $t = (t_B, t_N)$ avec :
  $t_B = 0, \quad t_N = D_{NB} l_B + D_{Nj0} l_{j_0}$
- **Mise à jour**
  $\bar{x} = x + \theta d, \quad \bar{E} = E + \theta t$

S4 : Changement de support

- **Si** $\theta = \theta_{j_0}$, Alors
  $\bar{J}_B = J_B, \quad \bar{J}_N = J_N$
- **Si** $\theta = \theta_{j_1}$, Alors
  $\bar{J}_B = J_B \backslash j_1, \quad \bar{J}_N = J_N \cup j_1$
- **Si** $\theta = \theta_F$ Alors
  $\bar{J}_B = J_B \cup j_0, \quad \bar{J}_N = J_N \backslash j_0$

S5 :Mise à jour $x = \bar{x}, \quad E = \bar{E}, \quad J_B = \bar{J}_B, \quad J_N = \bar{J}_N$

aller à S2

Fin.

3.8 Exemple numérique

Considérons le problème d'optimisation quadratique (3.1) à trois variables de décison, avec les données suivantes :

$$D = \begin{pmatrix} 4 & 2 & 1 \\ 2 & 3 & 0 \\ 1 & 0 & 2 \end{pmatrix}; \quad c = \begin{pmatrix} 1 \\ -2 \\ 2 \end{pmatrix}; \quad l = \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}; \quad u = \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix}.$$

Itération 1 : En démarrant du point réalisable $x = (0, 0, 1)'$ et le support initial $J_B = \emptyset$, nous devons d'abord calculer $E = Dx + c$:

$$E_N = E = \begin{pmatrix} 4 & 2 & 1 \\ 2 & 3 & 0 \\ 1 & 0 & 2 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ -2 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ -2 \\ 4 \end{pmatrix}$$

Commençons par l'étape 2 de l'algorithme. En utilisant les conditions d'optimalité (3.5), nous remarquons que seule la variable x_2 ne vérifie pas ces relations. Alors, $J_{NNO} = \{2\}$ et $j_0 = 2$.

- Calculons l'estimation de suboptimalité $\beta\{x, J_B\}$.

$$\beta(x, J_B) = \sum_{J \in J_{NN}, E_j > 0} E_j(x_j - l_j) + \sum_{j \in J_{NN}, E_j < 0} E_j(x_j - u_j)$$

$$= E_2(x_2 - u_2) = -2(0 - 4) = 8 > \epsilon.$$

Comme le critère de suboptimalité n'est pas vérifié, alors on exécute l'étape **(S3)** de l'algorithme.

- Calculons les directions $d = d_N = (0, 1, 0)'$ et $t = Dd = (2, 3, 0)'$
- Le pas optimal $\theta = \min\{\theta_{j_0}, \theta_{j_1}, \theta_F\}$, tels que :
  $\theta_{j_0} = 4 - 0 = 4; \quad \theta_{j_1} = \infty,$ car $J_B = \emptyset; \quad \theta_F = \frac{2}{3}$. Alors $\theta = \min\{4, \infty, \frac{2}{3}\} = \frac{2}{3} = \theta_F$.
- Mise à jour de la nouvelle SR $\bar{x}$ et son vecteur des coûts réduits $\bar{E}$.

$$\bar{x} = x + \theta d = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + \frac{2}{3} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{2}{3} \\ 1 \end{pmatrix}; \qquad \bar{E} = E + \theta t = \begin{pmatrix} 2 \\ -2 \\ 4 \end{pmatrix} + \frac{2}{3} \begin{pmatrix} 2 \\ 3 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{10}{3} \\ 0 \\ 4 \end{pmatrix}.$$

Le changement de support : $\bar{J}_B = \{2\}, \quad \bar{J}_N = \{1, 3\}$.

Itération 2 :

Pour continuer cette itération, nous utilisons les nouvelles valeurs mise à jour obtenues à la fin de l'itération prècédente :

$$\bar{x} = \begin{pmatrix} 0 \\ \frac{2}{3} \\ 1 \end{pmatrix}; \qquad \bar{E} = \begin{pmatrix} \frac{10}{3} \\ 0 \\ 4 \end{pmatrix}$$

— Calculons la nouvelle valeur d'estimation de suboptimalité $\beta(\bar{x}, \bar{J}_B)$

$$\beta(\bar{x}, \bar{J}_B) = \sum_{J \in J_{NN}, E_j > 0} E_j(x_j - l_j) + \sum_{j \in J_{NN}, E_j < 0} E_j(x_j - u_j)$$

$$= E_2(x_2 - l_2) + E_2(x_2 - u_2) = 0(\frac{2}{3} - (-1)) + 0(\frac{2}{3} - 4) = 0$$

Comme $\beta(\bar{x}, \bar{J}_B)$, étant nulle, le critère d'optimalité est vérifié.

D'où $x^0 = \begin{pmatrix} 0 \\ \frac{2}{3} \\ 1 \end{pmatrix}$ est une solution optimale.

3.9 Conclusion

Dans ce chapitre, nous avons vu certaines définitions dont nous avons besoin dans les étapes de la méthode support, l'algorithme de résolution par la méthode ,et enfin nous avons fourni un exemple numérique .

Chapitre 4

Application de la méthode de support aux problèmes SVMs

Dans ce chapitre, nous présenterons les résultats numériques de la méthode support présentée dans le chapitre précédent, implémentée en Python, puis nous l'adaptons aux problèmes SVMs. Ensuite, nous comparons numériquement la méthode de support et la méthode d'Optimisation Minimale Séquentielle. Cette comparaison est effectuée sur des problèmes de minimisation quadratique convexe générés aléatoirement.

4.1 Exemple d'exécution

Nous avons déjà trouvé la solution du problème(3.1), et maintenant nous allons résoudre ce problème sous PYTHON par l'algorithme implémenté.Voici l'appel de ce programmme :

$$x_{\text{final}}, J_{\text{B\_final}}, J_{\text{N\_final}}, \text{iterations}, \text{execution\_time}, \text{value obj. fun} =$$

$$\text{pqc\_algorithm}(D, c, l, u, \epsilon, \text{max\_iterations})$$

4.1.1 Les paramètres d'E/S de la l'algorithme de la méthode direct de support :

Les paramétres d'entrées :

D, c, l, u, ϵ, max_iterations

Les paramètres de sortie :

x_{final} : est le plan optimal.
$J_{\text{B\_final}}$: est l'ensemble des indices des variables basique.
$J_{\text{N\_final}}$: est l'ensembre des indices des variables non basique
iterations : est le nombre d'itérations du programme jusqu'à l'obtention du plan minimal.
execution_time : est la durée de l'exécution du programme.
value obj.fun : est la valeur de la fonction objectif.

```
D: [[4 2 1]
 [2 3 0]
 [1 0 2]]
c: [ 1 -2  2]
l: [ 0 -1  1]
u: [3 4 5]
```

```
Iteration 1
x: [ 0 -1  1]
E: [ 0 -5  4]
Beta: 25
Non-basic indices (J_NNO): [1]
Chosen j0: 1
Direction d: [0. 1. 0.]
t: [2. 3. 0.]
Updated x: [0.         0.66666667 1.        ]
Updated E: [3.33333333 0.         4.        ]
J_B: [1]
J_N: [0, 2]

Iteration 2
x: [0.         0.66666667 1.        ]
E: [3.33333333 0.         4.        ]
Beta: 0.0
Optimal solution found.

Final solution: [0.         0.66666667 1.        ]

 Value Obj. Fun= 2.333333333333333
Final basis set: [1]
Final no basis set: [0, 2]
Number of iterations: 2
Execution time (seconds): 0.01292729377746582
```

4.2 Implémentation

4.2.1 Langage de Programmation utilisé :

Python :

Dans ce projet, nous avons choisi d'utiliser le langage de programmation Python. Python est un langage informatique polyvalent largement utilisé pour diverses tâches telles que l'analyse et le traitement des données, ainsi que pour le machine learning et le deep learning. Il bénéficie d'une vaste collection de bibliothèques puissantes et pratiques.

FIGURE 4.1 – Logo Python.

4.2.2 Environnement de développement :

Google Colab :

Google Colaboratory est un outil d'analyse de données et d'apprentissage automatique qui vous permet de combiner du code Python exécutable et du texte enrichi avec des graphiques, des images, HTML, LaTeX et plus encore dans un seul document stocké dans Google Drive. Il se connecte aux puissants environnements d'exécution de Google Cloud Platform et vous permet de partager facilement votre travail et de collaborer avec d'autres.

FIGURE 4.2 – Google colab.

4.2.3 Les bibliothèques utilisé :

Choix	**Utilisation**	**Remarques**
Pandas	analyse et manipulation des données	- outil d'analyse et de manipulation des données open source, construit sur la base du langage de programmation Python.
NumPy	manipule les vecteurs et les matrices (tableaux multidimensionnels)	- outils mathématique utilisé pour effectuer différentes opérations mathématiques sur les tableaux.
SVC	bibliothèque de machine learning	- Partie de Scikit-learn utilisée pour la classification de données en utilisant les machines à vecteurs de support (SVM) - Permet de séparer les données en classes distinctes par l'utilisation de marges maximales.
Matplotlib	Tracer et visualiser les données sous formes de graphiques	- Outil d'analyse de données. Il peut être combiné avec les bibliothèques python de calcul scientifique NumPy et SciPy.
GridSearchCV	Technique utilisée en apprentissage automatique	- Recherche des meilleurs hyperparamètres pour un modèle d'apprentissage automatique. - Èvaluer les performances du modèle.
Scikit-learn	bibliothèque de machine learning	- Outils pour la classification, la régression, le clustering, et la réduction de dimensionnalité.

TABLE 4.1 – Langage de programation et bibliothèques utilisés

4.3 Méthode de support pour la résolution des problèmes SVM

Pour déterminer les paramètres optimaux de l'hyperplan séparateur d'un classifieur SVM, il est nécessaire de résoudre le programme quadratique convexe suivant, qui est associé à la formulation duale du problème SVM :

$$\begin{array}{c} \max\limits_{\alpha} -\frac{1}{2}\alpha^{'}\bar{Q}\alpha + e^{'}\alpha, \\ 0 \leq \alpha \leq Ce, \end{array} \tag{4.1}$$

où α est un n-vecteur et e est également un n-vecteur dont chaque composante est égale à 1. La matrice de Gram-Schmidt $D_{ij} = y_i y_j k(x_i, x_j)$ est une matrice carrée d'ordre n, symétrique, et semi-définie positive lorsque le noyau k vérifie les conditions de Mercer [20].
En comparant la formulation du problème pqsvm par la formulation du problème (3.1),que nous avons vu dans le chapitre précédent, on trouve :

$$\begin{cases} \bar{Q} = D, & \alpha = x, \qquad -e = c \\ C_e = u, & l = (0, ..., 0) \end{cases}$$

Alors, la formulation du problème SVM est adéquate pour appliquer la méthode support avec la particularité $|J_B| = 0$. En effet, on a une seule contrainte de variables bornées. Donc, l'application de la méthode support sur cette formulation est plus simple que la formulation standard de la $P.Q.C$.

4.4 Comparaison entre la méthode MDS et SMO

Dans cette section, nous appliquons notre approche, appelée Méthode directe de support (MDS), pour un problème quadratique convexe à variables bornées, à la résolution de problèmes de classification binaire par SVM. Nous l'avons implémentée dans le langage de programmation Python, sur un PC portable HP sous système d'exploitation Windows 11, équipé d'un processeur avec une fréquence de 1.90 GHz et d'une RAM de 8 Go.
Nous comparons notre méthode avec Optimisation Minimale Séquentielle (SMO), disponible dans la bibliothèque scikit-learn pour les SVM en Python. Nous utilisons le noyau RBF (Radial Basis Function) défini par : $k(x_i, x_j) = e^{-\sigma(||x_i - x_j||)^2}$ où σ est une constante positive.

Critères de comparaison :

- **CPU** : Le temps d'exécution des problèmes tests en secondes. On a utilisé le module time avec une fonction décorateur.
- **nit** : Le nombre d'itérations requis pour la résolution des problèmes tests.
- **ns** : Le nombre de vecteurs supports .
- **L'accuracy** : (exactitude) est une métrique de performance fréquemment employée pour évaluer l'efficacité des modèles de classification. Elle indique la proportion d'observations que le modèle a classifié correctement.Elle est définie par la formule suivante :

$$\text{Accuracy} = \frac{\text{Nombre de prédictions correctes}}{\text{Nombre total de prédictions}} \tag{4.2}$$

4.4.1 Les bases de données utilisées :

Avant de passer aux descriptions des bases de données utilisées [40], nous présentons un tableau récapitulatif des ensembles de données, incluant la taille du dataset, le nombre d'attributs, et les paramètres optimaux 'gamma' et 'C' choisis par recherche en grille.

Dataset	n	p	Gamma	C
Fruit	59	7	0.1	1
Iris	150	6	0.25	100
Heart	303	14	1	10
Breast-cancer	569	31	0.5	10
Diabète	768	9	1	1000
German	1000	20	0.03	1000
Digit	1797	64	0.01	1000

TABLE 4.2 – Description des bases de données utilisées

- **Fruit :** L'ensemble de données sur les fruits (59 exemples ayant 7 caractéristiques) comprend des détails sur diverses propriétés des différentes variétés de fruits, telles que leurs masse, taille et la couleur. Dans notre projet, nous avons exploité cet ensemble de données pour différencier les *mandarines* des *pommes*.
- **Iris :**Iris : L'ensemble des données sur les iris de Fisher est composé de 150 exemples, chacun possédant 6 attributs numériques. Cette base représente trois espèces de fleurs distinctes : Iris setosa, Iris virginica et Iris versicolor. Pour notre projet, nous avons réduit ce problème à une classification binaire en sélectionnant uniquement deux classes : Iris **setosa** et Iris **versicolor**. Cette simplification nous a permis de développer des modèles de classification visant à différencier efficacement ces deux espèces d'iris en se basant sur leurs caractéristiques mesurées. En utilisant ces données, nous avons pu concevoir et évaluer des algorithmes de classification capables de distinguer avec précision entre les iris setosa et versicolor en se basant sur les mesures de leurs sépales et pétales.
- **Heart :** Le jeu de données sur le cœur, avec ses 303 échantillons et 14 attributs, est d'une grande valeur dans le domaine médical et de la recherche cardiovasculaire. Dans notre travail, nous avons exploité ces données pour prédire la présence ou l'absence de maladies cardiaques chez un individu. En analysant les relations entre les attributs tels que le sexe, la tension artérielle, le cholestérol sérique et d'autres mesures physiologiques.
- **Breast-cancer :** Le jeu de données sur le cancer du sein, avec ses 569 échantillons et 31 attributs, est une ressource cruciale pour la recherche médicale. Il contient des données cliniques et histologiques essentielles sur les tumeurs mammaires, ainsi que des informations sur les patients. Dans notre projet, nous avons utilisé ce dataset dans le but de prédire si une tumeur est bénigne ou maligne. En exploitant les caractéristiques extraites des données cliniques et histologiques.
- **Diabète :** Dans notre étude, nous avons utilisé l'ensemble de données sur le diabète, qui comprend 768 échantillons avec 9 attributs distincts et 2 classes. Ces données ont été créées à l'origine par l'Institut national du diabète et des maladies digestives et rénales. Notre objectif était de prédire de manière diagnostique si un patient est diabétique ou non, en nous appuyant sur des mesures diagnostiques spécifiques incluses dans cet ensemble de données.

- **German :** Comprend 1000 échantillons et 20 attributs, ce qui en fait une ressource précieuse pour l'apprentissage automatique. Dans notre travail, nous avons utilisé pour prédire si une demande de crédit devrait être approuvée ou non pour un individu en se basant sur ses caractéristiques financières. C'est une application importante de la classification dans le domaine financier, qui aide les institutions à prendre des décisions éclairées sur l'octroi de crédit et à gérer les risques financiers associés.
- **Digit :** Le jeu de données "digits" contient 1797 échantillons d'images de chiffres manuscrits de taille 8x8. Ce jeu de données est couramment utilisé dans le domaine de l'apprentissage automatique pour la classification d'images de chiffres allant de 0 à 9. Dans le cadre de notre travail, nous avons transformé ce problème de classification en un problème binaire. Plus précisément, nous avons regroupé les chiffres pairs (0, 2, 4, 6, 8) sous l'étiquette -1, et les chiffres impairs (1, 3, 5, 7, 9) sous l'étiquette 1. L'objectif de notre projet est donc de développer et d'évaluer des modèles de classification capables de différencier efficacement entre ces deux classes (pairs et impairs) en se basant sur les caractéristiques des images de chiffres.

4.5 Discussion des résultats :

	MDS				**SMO**			
Base	**ns**	**CPU**	**nit**	**Acc**	**ns**	**CPU**	**nit**	**Acc**
Fruit	7	0.020	16	74	7	0.001	58	100
Iris	33	0.026	33	100	30	0.001	82	100
Heart	299	0.783	300	84	300	0.009	659	100
Breast-cancer	526	4.420	529	94	526	0.036	656	100
Diabète	622	6.596	634	73	620	0.094	1540	100
German	793	18.788	803	72	789	0.131	2193	100
Digit	298	4.873	349	83	298	0.074	2206	100

TABLE 4.3 – Les résultats de comparaison entre la méthode directe de support et SMO.

Le tableau (Table 2) présente les résultats comparatifs des deux méthodes sur différentes bases de données sous Python. Les critères de comparaison sont le temps d'exécution, le nombre d'itérations, ainsi que le nombre de vecteurs supports.

D'après le tableau, on remarque que la méthode SMO est plus rapide que MDS pour converger vers une solution optimale. Cette différence est particulièrement marquée pour des ensembles de données plus grands comme Digits. En termes de nombre d'itérations, la méthode SMO nécessite plus d'itérations que MDS pour converger malgré un temps d'exécution global plus rapide. Cela peut s'expliquer par l'efficacité de MDS à traiter chaque itération plus rapidement en décomposant le problème global en sous-problèmes plus simples.

Les deux méthodes produisent des nombres de vecteurs supports similaires, bien que légèrement différents. Une particularité de l'algorithme MDS est qu'il utilise l'estimation de sub-optimalité, permettant d'arrêter l'algorithme avec une précision désirée et d'améliorer ainsi la convergence.

En termes de précision, la méthode SMO atteint systématiquement un taux de 100% sur toutes les bases de données testées, tandis que la méthode MDS présente des variations. Par exemple, sur la base de données "fruit", la précision de MDS est de 74%, contre 100% pour

SMO. Cela démontre que la méthode SMO non seulement converge plus rapidement, mais aussi avec une précision égale ou supérieure dans tous les cas testés.

4.5.1 Visualisation

Dans cette partie, nous présentons visuellement l'hyperplan généré par la méthode MDS, sur les données Iris :

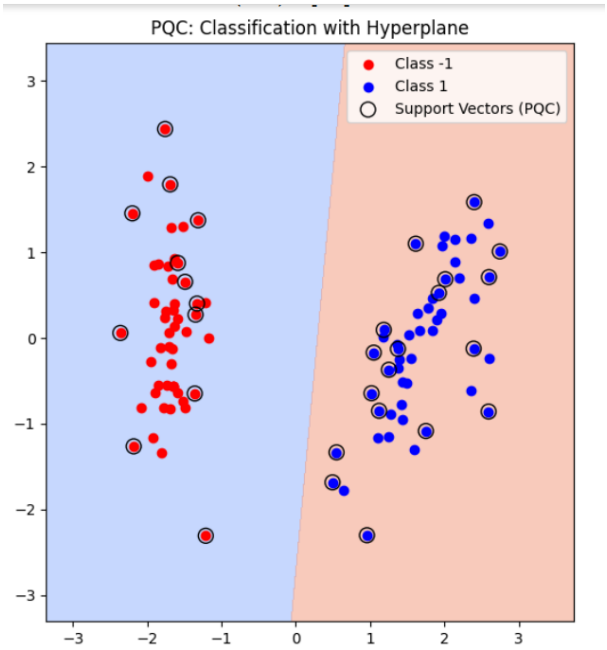

FIGURE 4.3 – Hyperplan de classification de la MDS sur les données Iris.

4.6 Conclusion

Dans ce chapitre, nous présenterons les résultats numériques de la méthode de support introduite dans le chapitre précédent et implémentée en Python. Nous avons effectué une comparaison numérique entre la méthode de support et la méthode d'Optimisation Minimale Séquentielle, toutes deux implémentées en Python. Cette comparaison a été réalisée sur des problèmes de minimisation quadratique convexe générés aléatoirement.

Conclusion Générale

L'objectif principal de ce mémoire est de proposer une nouvelle méthode pratique et efficace de résolution des problèmes de SVM régularisés. L'approche proposée est appliquée aux problèmes dual associée à un SVM type L2-régularisée, avec une fonction de pertre basé sur la norme L1. L'algorithme de la MDS de résolution d'un PQ convexe a variables bornées et présenter, puis illustrer par un exemple didactique. Puis comparer ses performances avec la méthode SMO (Optimisation Minimale Séquentielle), en termes de temps d'exécution, de nombre d'itérations, de la précision et le nombre de vecteur support, puis d'implémenter MDS en Python. Les exprémentation numériques sur septs benchemarks de UCI [40] ont montré l'applicabilité de la MDS à la résolution de ce type de problèmes. Pour atteindre ces objectifs, nous avons suivi une démarche structurée comme suite .

Dans le premier chapitre, nous avons posé les bases théoriques en introduisant les concepts fondamentaux de l'optimisation, notamment la convexité, les formes quadratiques et les conditions d'optimalité. Nous avons également exploré la dualité lagrangienne et ses applications en optimisation quadratique convexe.

Le deuxième chapitre est totalemnt dédie aux problèmes de calsse binaire par SVM,tout en détaillant leur formulation mathématique, les fonctions noyaux et les techniques pour traiter les problèmes de classification linéaires et non linéaires. Nous avons mis en évidence l'importance des SVM dans divers domaines d'application, soulignant leur rôle crucial dans l'apprentissage automatique supervisé.

Dans le troisième chapitre, nous avons développé et illustré la méthode de support pour la minimisation des problèmes quadratiques convexes à variables bornées. Nous avons présenté un algorithme détaillé basé sur cette méthode, accompagné d'un exemple numérique démontrant ces différentes étapes.

Le quatrième chapitre a été consacré à l'implémentation de la méthode de support sur Python et à son application aux SVMs régularisés. Nous avons comparé notre approche avec celle de SMO en utilisant septs bases d'apprentissage de UCI [40] des problèmes générés aléatoirement. Les résultats ont montré que notre approche offre des avantages significatifs en termes de flexibilité et de convergence rapide.

En conclusion, ce travil a démontré l'efficacité de la méthode de support pour résoudre les problèmes quadratiques convexes à variables bornées pratique dans applications aux SVMs ré-

gularisés. L'implémentation de la MDS en Python à permis de valider l'aproche proposée et a donné des résultats numériques solides. La comparaison avec la méthode SMO a mis en évidence les avantages et les limites de chaque méthode, offrant ainsi des perspectives précieuses pour de futures applications.

Comme perspectives, nous proposons les travaux futurs de recherche suivants :

- Pour améliorer le temps d'exécution, il serait intéressant d'explorer le recalcul de la matrice inverse afin de réduire le temps global de calcul. Cette optimisation pourrait potentiellement accélérer la convergence de l'algorithme, surtout pour les ensembles de données de grande taille.
- Application de cette méthode pour les problèmes d'apprentissage automatique, tels que la classification multi-classes, la régression et le clustering.

Bibliographie

[1] R. Gabasov, and F. M. Kirillova," Methods of Linear Programming," Edition of the Minsk University, vol. 1,2 and 3 , 1977,1978 and 1980.

[2] E. Osuna, R. Freund, and F. Girosi," Training support vector machines : an application to face detection," In : the IEEE Conference on Computer Vision and Pattern Recognition(CVPR' 97), pp. 130-136, 1997.

[3] V. N. Vapnik," Estimation of dependences based on empirical data," Springer Series in Statistics. Springer-Verlag, New York, 1982.

[4] C. Platt," Fast training of support vector machines using sequential minimal optimization," In : B.Schlkopf, C.C. Burges, and A.J. Smola, editors, Advances in Kernel Methods : Support Vector Learning, MIT Press, pp. 185-208, 1999.

[5] K. Scheinberg," An efficient implementation og an active set method for svms," J. Mach. Learn. Res, Vol. 7, pp. 2237-2257, 2006.

[6] JP. Vert," support vector machines in bioinformatics," Human Genome Center University, tokyo, JAPAN, July 17-19, 2002.

[7] S. Mokhtar, Bazarra. hanif D.Sherali.C.M.Shetty.non linéaire programmatique theory and algorithme, pp. 40-41, 2006.

[8] A. Djeffal, cours support vector machine. Récupéré sur site personnel abdelhamid, 2012.

[9] V. Vapnik," The nature of Statistical Learning Theory," Information Science and Statistics, Springer-verlag, New York, 1995.

[10] V.N. Vapnik. The nature of statistical learning theory. Springer, 2000.

[11] document en suivant ce lien : Machines à Vecteurs Supports Didacticiel.

[12] H. Mohamadally, and Fomani Boris," SVM machine à vecteurs de support ou séparateur à vaste marge," BD Web, ISTY3,Versailles St Quentin, France, janvier 2006.

[13] N. Ikheneche," Méthode de support pour la minimisation d'une fonctionnelle quadratique convexe," Mémoire de Magister. Département de Recherche Opérationnelle, Université de Béjaia, 2004.

[14] M. Bentoubache," Sur les méthodes mathématiques de la programmation linéaire et quadratique," PhD thesis. Université de Béjaia, 2013.

[15] B. Schlkopf and A. Smola," Learning with kernels : support vector machines, regularization, optimization and beyond," MIT Press Cambridge, MA, USA, pp. 1-24, 2001.

[16] C. Sentelle, G. Anagnostopoulos, and M. Georgiopoulos," Efficient revised simplex method for svm training," IEEE Trans. Neural Netw, Vol. 22, pp. 1650-1661, 2011.

[17] V. Vapnik," Statistical learning theory," Edition Wiley, 1998.

[18] E. Byvatov, and G. Schneider," Support vector machine applications in bioinformatics," Appl. Bioinformatics, vol. 2, pp. 67-77, 2003.

[19] C. Burges," A tutorial on support vector machines for pattern recognition," Data Min.Knowl. Discov, vol. 2, pp. 121-167, 1998.

[20] J. Mercer," Functions of positive and negative type, and their connection with the theory of integral equations," Philosophical Transactions of the Royal Society, London, A, 209, pp. 415-446, 1909.

[21] B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, Advances in Kernel Methods - Support Vector Learning, Cambridge, MA, MIT Press, 1998.

[22] A. Djeffal, M. Babahenini, and A. Taleb-Ahmed," A fast multi-class svm learning method for huge databases," International Journal of Computer Science Issues, vol.8, pp. 544-550, 2011.

[23] B. Brahmi," Méthodes primales et duales pour la programmation quadratique : extension et applications," Thèse de doctorat en mathématiques appliquées, université de Béjaia, 2012.

[24] P. Dussault," Optimisation mathématique avec applications en imagerie," pp. 20, 2017.

[25] A. Iouditski," Optimisation : Analyse convexe et théorie de programmation non-linéaire," Note de cours 1ère année magistère de mathématique, Univ Joseph Fourier, 2007.

[26] O. L. Mangasarian," Nonlinear Programming," McGraw-Hill, New York, 1969.

[27] M. O. Bibi," Téchniques numériques d'optimisation," Cours de Master 2 en Mathématiques financières, 2019.

[28] A. ANDJOUH , "Minimisation d'une forme quadratique avec une ou plusieurs valeurs propres négatives," Thèse de doctorat en mathématiques appliquées, université de Béjaia, 2024.

[29] H. J. Ferreau," An online active set strategy for fast solution of parametric quadratic programs with applications to predictive engine control," PhD thesis, university of Heidelberg, 2006.

[30] K. J. Arrow, and G. Debreu," Existence of an equilibrium for a competitive economy," Econometrica, vol. 22, pp. 265–290, 1954.

[31] S. Chelouti, and K. Kaidi," Résolution d'un problème d'optimisation multi-objectif fractionnaire Linéaire flou en nombres entiers," Mémoire de Master, Université de Boumerdes, 2016.

[32] V. Vapnik, S. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing," The Advances in Neural Information Processing Systems, MIT Press, pp. 281-287, 1997.

[33] L. El-Naqa, Y.Yang, M.N.Wernick, N. Galatsanos, and R. M. Nishikawa, "A support vector machine approach for detection of microcalcifications," IEEE Transactions on Medical Imaging, vol. 21, pp. 1552-1563, 2002.

[34] M. Vogt, U. Moissl, and J. Schaab," Heart rate classification using support vector machines," From Data and Information Analysis to Knowledge Engineering, springer, pp. 716- 723, 2006.

[35] B. Krishnapuram, J. Sichina, and L. Carin, " Physics-based detection of targets in sar imagery using support vector machines," IEEE Sensors Journal, vol. 3, pp. 147-157, 2003.

[36] X. Wang, P. Phua, and W. Lin," Stock market prediction using neural networks : Does trading volume help in short-term prediction ?," In Neural Networks, Proceedings of the International Joint Conference on, vol. 4, pp. 2438-2442, 2003.

[37] B. Brahmi,"Apprentissage Statistique et Optimisation," Note de cours 2ème année master SDAD , université de Béjaia, 2023.

[38] F. E. Tay, and L. Cao," Application of support vector machines in financial time series forecasting," Omega : The International Journal of Management Science, 2001.

[39] T. V. Gestel, J. Suykens, and D.-E. B. et al," Financial time series prediction using least squares support vector machines within the evidence framework,"IEEE Trans. Neural Netw, vol. 12, pp. 809-821, 2001.

[40] A. Asuncion, D. Newman, Uci machine learning repository, University of Ca li-fornia, School of Information and Computer Sciences, Irvine, CA, 2010, http :// dx.doi.org/http ://www.ics.uci.edu/mlearn/MLRepository.html.

Résumé

Dans ce mémoire, nous avons élaboré une nouvelle approche basée sur la méthode directe de support (MDS) pour la minimisation d'une fonction quadratique convexe à variables bornées où la matrice associée est semi-définie positive. La MDS est appliquée aux problèmes de SVM régularisés. Elle est particulièrement utile lorsque le problème PQ comporte un grand nombre de variables.Le principe de cette méthode est simple : partant d'une solution réalisable de support initiale, chaque itération consiste à trouver une direction d'amélioration et un pas maximal le long de cette direction en améliorant la valeur de la fonction objectif tout en veillant à ne pas sortir du domaine réalisable déterminé par les bornes du problème. Afin de comparer son efficacité avec la méthode SMO, nous avons implémenté la MDS sur Python. Les expérimentations numériques sur des benchmarks ont montré l'efficacité de notre méthode par rapport à SMO en termes de nombre d'itérations, mais en termes de temps CPU, c'est l'inverse qui se produit.

Mots-clés : Programmation quadratique convexe, Support vecteur machine (SVM), Méthode directe de support, Estimation de suboptimalité, Python.

Abstruct

In this report, we have developed a new approach based on the direct support method (DSM) for the minimization of a convex quadratic function with bounded variables where the associated matrix is positive semidefinite. MDS is applied to regularized SVM problems. It is particularly useful when the PQ problem has a large number of variables.The principle of this method is simple : starting from a feasible solution of initial support, each iteration consists in finding a direction of improvement and a maximum step along this direction by improving the value of the objective function, while taking care not to leave the feasible domain determined by the bounds of the problem. To compare its effectiveness with the SMO method, we implemented the MDS on Python. Numerical experiments on benchmarks have shown that our method is more efficient than SMO in terms of number of iterations, but in terms of CPU time, the opposite is true.

Key-words : Convex quadratic programming, Machine vector support (SVM), Direct support method, Suboptimality estimate, Python.