RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ ABDELHAMID IBN BADIS-MOSTAGANEM
FACULTÉ DES SCIENCES EXACTES ET DE L'INFORMATIQUE

DÉPARTEMENT DE MATHÉMATIQUES ET INFORMATIQUE

Mémoire de fin d'étude

Pour l'obtention du diplôme de Master en Mathématiques
Cycle LMD

OPTION : Modélisation, Contrôle et Optimisation

Présenté par

Melle Kaim Houaria

Soutenu le 21 Mai 2017

Intitulé

Optimisation Sequentielle Minimal (SMO) Pour Les Machines à Vecteurs Support (SVMs)

Devant le Jury

Hocine ABLAOUI	**Président**	MAA	**U. MOSTAGANEM**
Rachid BELGACEM	**Examinateur**	MAA	**U. Chlef**
Abdessamad AMIR	**Encadreur**	MCA	**U. MOSTAGANEM**

Année universitaire 2016/ 2017

Dédicaces

Tout d'abord, je veux rendre grâce à Dieu, le Clément et le Très Miséricordieux pour son amour éternel. C'est ainsi que je dédie ce mémoire à :
ma mère pour sa tendresse et mon père pour sa patience et encouragement
mes très chers frères et leurs familles et ma chère soeur et sa famille pour leurs conseils,
mes cousins et cousines,
tous ceux que j'aime,
tous mes amies.

Remerciements

Je remercie ALLAH de m'avoir donné la volonté et le courage qui m'ont permis de réaliser ce travail. Veuille t-Il me guider dans le droit chemin.
J'aimerai spécialement remercier Mer AMIR Abdssamad qui m'a proposé ce sujet et pour sa grande disponibilité pendant toute la durée de ce travail, pour m'avoir toujours encouragé face à la difféculté, mais aussi pour sa gentillesse .
Je remercie aussi les membres de jury, Mer Rachid BELGACEM et Mer ABLAOUI Hocine pour avoir accepter de juger mon travail.
Je remercie toute personne ayant participé de près ou de loin pour la réalisation de ce travail.
Finalement, je tiens à exprimer tous mes remerciements et ma profonde gratitude à toute ma famille..

Résumé

Les machines à vecteurs de support sont un ensemble de techniques d'apprentissage destinées à résoudre des problèmes de discrimination, c'est-à-dire décidé à quelle classe appartient un échantillon. Les SVM sont une généralisation des classifier linéaires. Dons ce mémoire propose un nouvel algorithme pour la formation de machines à vecteurs de support : Optimisation Minimal Séquentielle, ou SMO. L'apprentissage d'une machine à vecteur de support nécessite la solution de un très grand problème d'optimisation de programmation quadratique (QP). SMO casse cette grande Problème QP dans une série de problèmes QP les plus petits possibles. Ces petits problèmes de QP sont résolus analytiquement, ce qui évite d'utiliser une optimisation QP numérique longue en tant que Boucle interne.

Table des matières

1 Machines à vecteurs de support **7**
1.1 Le cas linéairement séparable 7
1.1.1 La Marge 9
1.1.2 L'hyperplan 10
1.1.3 Représentation duale 11
1.1.4 Vecteurs de support 11
1.2 Le cas non linéairement séparable 12
1.2.1 Marge souple 12
1.2.2 Représentation duale 13
1.2.3 Les noyaux 14

2 La méthode Active-set **17**
2.1 Description de la méthode 17
2.2 Algorithme Active-Set 19
2.2.1 Exemple 20

3 Optimisation Minimal Séquentielle **23**
3.1 Introduction 23
3.2 La méthode Analytique 24
3.3 Heuristiques pour le choix des deux multiplicateurs 25
3.4 Le calcul de b 26

4 Experimentations numériques **27**

5 Conclusion **28**

Bibliographie **29**

Introduction

Les machines à vecteurs de support ou séparateurs à vaste marge (en anglais Support Vector Machine, SVM) initialement développées par Vapnik, sont un ensemble de techniques d'apprentissage destinées à résoudre des problèmes de classification ou de régression. Ces techniques ont été appliquées dans divers domaines, à savoir la reconnaissance des caractères, la reconnaissance d'images et le diagnostique médical ...etc. Dans la plupart de ces applications, le pouvoir de généralisation des SVM est bien meilleur que d'autres méthodes compétitives. Pour le cas de la classification binaire, où les données d'apprentissage viennent uniquement de deux classes différentes, la recherche d'un hyperplan séparateur entre les deux classes de l'échantillon d'apprentissage revient à résoudre un programme quadratique convexe. Vue la grande taille des problèmes SVMs posés dans la pratique, les méthodes classiques d'optimisation ne permettent pas de les résoudre, ce qui nécessite de développer des approches spécifiques pour ce type de problèmes. Actuellement, les approches de décomposition et les méthodes des contraintes actives sont les plus utilisées. Nous proposons dans ce travail deux méthodes, la première est la méthode Active Set et la deuxième est la méthode appelé Optimisation minimale séquentielle (SMO). La méthode Active-Set peut être vue comme une généralisation de la méthode du Simplexe de Dantzing, elle a été utilisée à partir des années 70, elle est efficace pour résoudre des problèmes quadratique de taille petite et moyenne. On propose après une nouvelle méthode conçue pour résoudre les SVMs, l'algorithme d'optimisation séquentielle optimale (SMO) a été proposé premièrement par Platt et all en 1999, c'est l'algorithme le plus utilisé actuellement pour les problèmes de grande taille. Les méthodes classiques de programmation quadratique nécessitent la résolution d'un très grand nombre de programme quadratique (QP) par fois de taille importante. SMO, brise ce gros problème en résolvant à chaque itération les plus petits programme QP possibles. Ces petits problèmes QP sont résolus de façon analytique, ce qui évite d'utiliser un solveur de QP en boucle interne. La quantité de mémoire requise pour SMO est linéaire dans la taille des données d'apprentissage, ce qui permet à SMO de gérer des ensembles d'apprentissage de taille très importante. Le présent mémoire est organisé en quatre chapitres. Le chapitre 1, décrit en détail le concept ainsi que la formulation mathématique des SVMs. Le deuxième chapitre est dédié à la méthode Active -Set comme méthode classique pour la programmation quadratique. Dans le chapitre 3, nous découvrons une version simplifiée de la méthode SMO, la méthode présentée est inspiré du papier de Platt. Une comparaison numérique entre les deux méthodes présentées est donnée au chapitre 4.

Chapitre 1

Machines à vecteurs de support

Les machines à vecteurs de support (En anglais Support vectors machines (SVMs)) sont des algorithmes d'apprentissage automatique, conçus à résoudre les problèmes de classification et de régression. Dans ce travail on considère uniquement le problème de classification à deux classes ; un problème dans lequel on tente de déterminer la classe à laquelle appartient un individu parmi les m. Pour ce faire, on utilise les caractéristiques connues de cet individu. Ces n caractéristiques sont représentées par un vecteur $x \in \mathbb{R}^n$. La classe à laquelle appartient l'individu est représentée par $y_i \in \{-1, +1\}$.L'idée principale des SVMs, est de rechercher un hyperplan qui séparera le mieux ces deux classes. Si un tel hyperplan existe, on dira que les données sont linéairement séparables.

1.1 Le cas linéairement séparable

Etant donné un échantillon d'apprentissage (x_i, y_i), $i = 1...n$ où $y_i \in \{-1, +1\}$. Supposons aussi que ces données sont linéairement séparables, c'est-à-dire qu'il existe un hyperplan dans $\mathbb{R}^n$ tel que toutes les données appartenant à la classe $+1$ se retrouvent d'un côté de l'hyperplan alors que celles de la classe se -1 se situent de l'autre côté. L'hyperplan

séparateur est représenté par l'équation suivante :

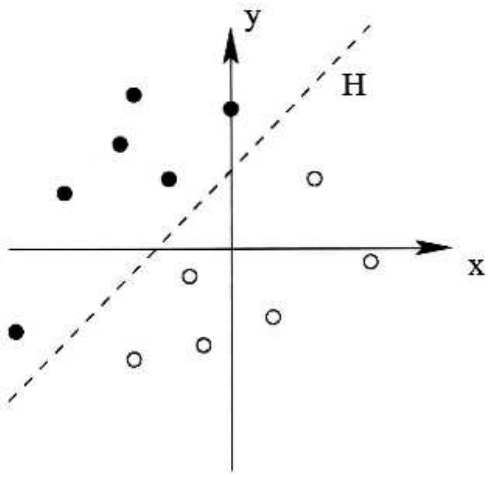


Figure 1 : Des donnes linairement sparables

$$w^T x + b \tag{1.1}$$

Puisque les deux classes sont linéairement séparables. Alors, il suffit d'utiliser la fonction suivante pour effectuer la classification :

$$\begin{cases} class = 1 \ si \ w^T x + b > 0 \\ class = -1 \ si \ w^T x + b < 0 \end{cases} \tag{1.2}$$

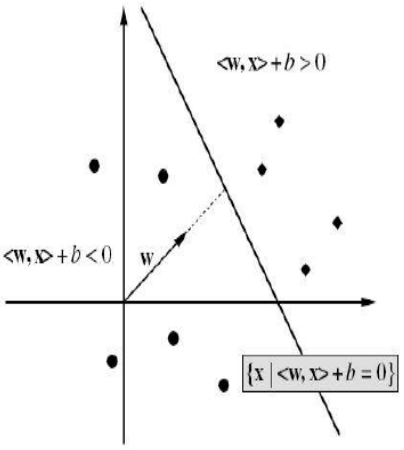


Figure 2 :Sparation par Hyperplan.

Cependant, si les données sont linéairement séparables, il existe une infinité d'hyperplans qui peuvent servir de séparateurs. Afin de déterminer ce qui caractérise le meilleur hyperplan, introduisons le concept de marge.

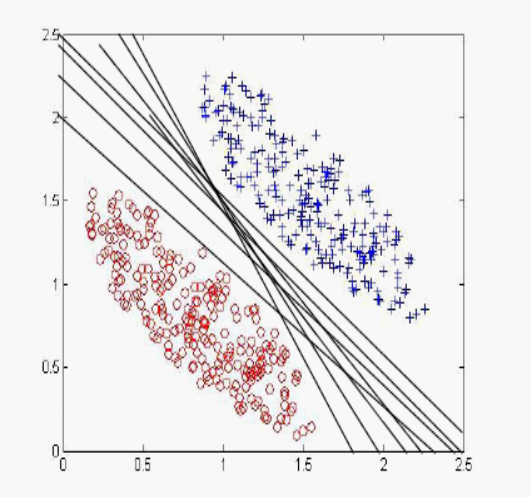


*Figure*3 :Il existe une infinit d'hyperplans pouvant sparer les donnes.

1.1.1 La Marge

Définissons la marge d'un hyperplan comme étant la distance entre l'hyperplan et la donnée la plus proche. Plus formellement, si $dist(x;(w,b))$ représente la distance euclidienne entre le point x et l'hyperplan $w^T x + b = 0$, alors la marge M est définie ainsi :

$$M = min\left\{dist(x_i;(w,b))\right\} : i = 1,\ldots,m,$$

où les x_i sont les données de l'ensemble d'apprentissage. L'hyperplan qui aura la meilleure généralisation est celui qui possèdera la plus grande marge (Vapnik),si les données sont linéairement séparables, les SVMs trouvent l'hyperplan qui sépare les données avec la plus vaste marge possible, puis utilisent cet hyperplan pour classer de nouvelles données à l'aide de la fonction de classification $f(x) = w^T x + b$.

1.1.2 L'hyperplan

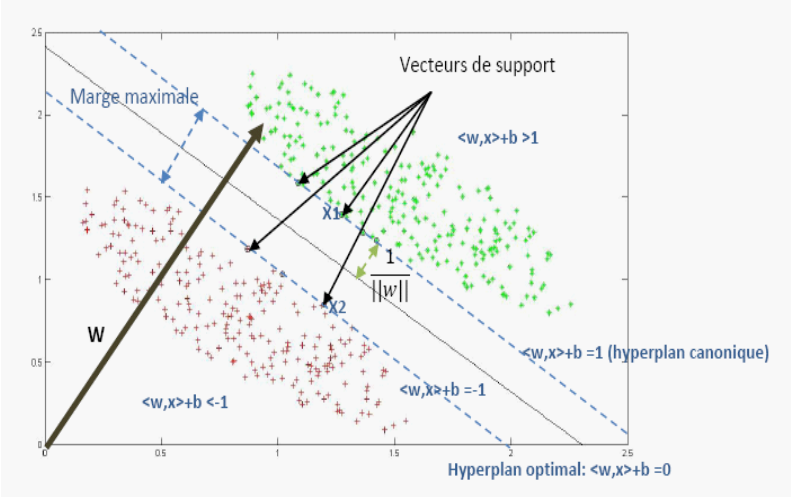


Figure 4 :hyperplan optimal,vecteurs de support et marge maximale.

L'hyperplan $w^T x + b = 0$ représente un hyperplan séparateur des deux classes, la distance entre cet hyperplan et l'exemple le plus proche s'appelle la marge. La distance entre l'exemple le plus proche et l'hyperplan est donnée par :

$$M = \frac{1}{\|w\|} \tag{1.3}$$

où $\|w\| = \sqrt{w_1^2 + w_2^2 + ...w_n^2}$. Puisque les données sont linéairement séparables, alors il existera un hyperplan d'équation $w^T x + b = 0$,tel que

$$\begin{aligned} w^T x_i + b &\geq 1, pour \quad y_i = 1. \\ w^T x_i + b &\leq -1, pour \quad y_i = -1. \end{aligned}$$

On peut combiner ces deux inéquations en une seule telle que :

$$y_i(w^T x_i + b) \geq 1.$$

La recherche du meilleur hyperplan peut donc s'écrire sous la forme d'un problème d'optimisation :

$$\begin{cases} \max \frac{2}{\|w\|} \\ y_i(w^T x_i + b) \geq 1 \end{cases} \tag{1.4}$$

$\Longleftrightarrow$

$$\begin{cases} \min \|w\| \\ y_i(w^T x_i + b) \geq 1 \end{cases}$$

$\Longleftrightarrow$

$$\begin{cases} \min \frac{1}{2} \|w\|^2 \\ y_i(w^T x_i + b) \geq 1 \end{cases}$$

1.1.3 Représentation duale

Il serait possible de résoudre le problème d'optimisation ci-dessus directement. Toutefois, sa représentation duale possède des propriétés très intéressantes qui auront des répercussions majeures lorsque nous considérerons les machines à vecteurs de supports pour le cas où les données ne sont pas linéairement séparables. Ce genre de problème d'optimisation peut être résolu en associant un multiplicateur de Lagrange α_i à chaque contrainte ($\alpha_i > 0$). Le Lagrangien est donné par :

$$L(w, \alpha, b) = \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i (y_i (w^T x_i + b) - 1) \tag{1.5}$$

En passant à la formulation duale, le problème devient : maximiser le Lagrangien, cela revient à dire, de trouver α_i, b et w qui annulent ses dérivées partielles :

$$\begin{cases} \frac{\partial L(w,\alpha,b)}{\partial w} = 0 \\ \frac{\partial L(w,\alpha,b)}{\partial b} = 0 \\ \alpha_i \geq 0 \end{cases} \tag{1.6}$$

On trouve :

$$\begin{aligned} w &= \sum_{i=1}^{n} \alpha_i y_i x_i \\ \sum_{i=1}^{n} \alpha_i y_i &= 0 \end{aligned} \tag{1.7}$$

En remplaçant 1.7 dans la fonction objective 1.5, on obtient le problème dual à maximiser suivant :

$$\begin{cases} \max L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \\ \alpha_i \geq 0 \end{cases} \tag{1.8}$$

Si le problème de classification est linéairement séparable, une solution optimale pour les α_i existe.

1.1.4 Vecteurs de support

Pour une tache de détermination de l'hyperplan séparable des SVM est d'utiliser seulement les points les plus proches (i.e. les points de la frontière entre les deux classes des données) parmi l'ensemble total d'apprentissage, ces point sont appelés vecteurs de support .La raison de ce nom est que ce sont les seuls points utiles pour déterminer l'hyperplan. Rappelons que le vecteur des coefficients de l'hyperplan est donné par :

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{1.9}$$

Ainsi, tout point qui n'est pas sur la marge n'apporte aucune contribution, puisque α_i est alors nul. Si tous les points sauf les vecteurs de support étaient retirés de l'ensemble d'apprentissage, on retrouverait le même hyperplan. Les vecteurs de support peuvent donc être vus comme les points contenant toute l'information essentielle du problème.

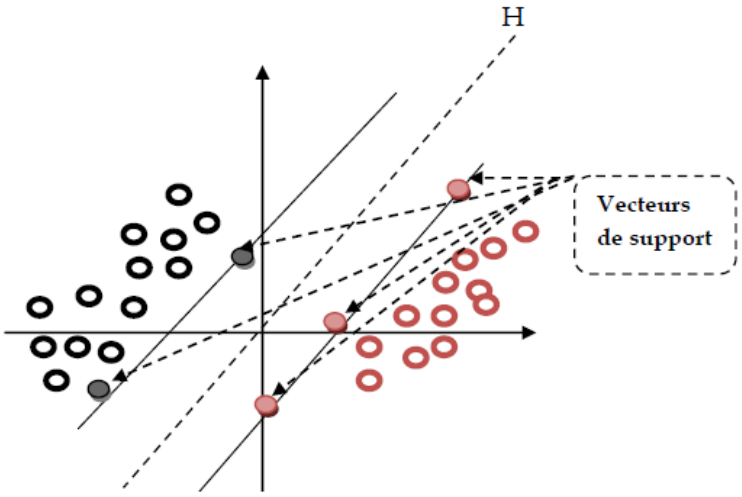


Figure 5 :Les vecteurs de support.

1.2 Le cas non linéairement séparable

En réalité, un hyperplan séparateur n'existe pas toujours, et même s'il existe, il ne représente pas généralement la meilleure solution pour la classification. En plus une erreur d'étiquetage dans les données d'entrainement (un exemple étiqueté $+1$au lieu de-1 par exemple) affectera crucialement l'hyperplan. Dans le cas où les données ne sont pas linéairement séparables, ou contiennent du bruit (Données mal étiquetées) les contraintes ne peuvent être vérifiées, et il y a nécessité de les relaxer un peu. Ceci peut être fait en admettant une certaine erreur de classification des données ce qui est appelé "SVM à marge souple (En anglais Soft Margin)".

1.2.1 Marge souple

Un meilleur moyen serait de permettre à quelques données d'être à l'intérieur de la marge ou du mauvais côté de l'hyperplan. Il s'agit du concept de marge souple (soft margin). Une première idée serait de tenter de maximiser la marge tout en minimisant le nombre de données mal classées. Toutefois, le nombre de données mal classées peut être trompeur, puisqu'il ne permet pas de déterminer si une donnée était presque correctement classée ou si elle était en réalité très loin de l'hyperplan. Une meilleure idée est d'attribuer à chaque donnée x_i une valeur $\xi_i$qui représente à quel point la donnée est éloignée d'un bon classement, puis tenter de minimiser la somme des ξ_i. Plus formellement, au lieu d'imposer. On introduit alors sur les contraintes des variables ξ_i dites de relaxation pour obtenir la contrainte de l'équation :

$$y_i(w^T x_i + b) \geq 1 - \xi_i \quad i = 1...n$$

Si $\xi_i < 1$, x_i ne respecte pas la marge mais reste bien classé, sinon x_i est mal classé par l'hyperplan. Dans ce cas, au lieu de rechercher uniquement un hyperplan séparateur qui maximise la marge, on recherche un hyperplan qui minimise aussi la somme des erreurs permises ξ_i . Le problème d'optimisation devient :

$$\begin{array}{c} \min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \\ y_i(w^T x_i + b) \geq 1 - \xi_i \; i = 1...n \\ \xi_i \geq 0 \end{array} \tag{1.10}$$

Où $C > 0$ est une constante qui représente la pénalité d'avoir des données mal classées. Lorsque C est très élevé, il y aura très peu de données mal classées.

1.2.2 Représentation duale

Il est possible de construire le dual de ce problème de la même manière que précédemment. Le Lagrangien est :

$$L(w, \alpha, \xi, \beta, b) = \frac{1}{2} w^T w - \sum_{i=1}^n \alpha_i (y_i(w^T x_i + b) - 1 + \xi) + C \sum_{i=1}^n \xi_i + \sum_{i=1}^n \beta_i \xi_i \tag{1.11}$$

où les α_i, β_i multiplicateurs de Lagrange. Afin de trouver le Lagrangien minimal pour un α_i, β_i donné, il faut le dériver par rapport aux variables primales. On obtient alors :

$$\frac{L(w, \alpha, \xi, \beta, b)}{\partial w} = 0 \Longrightarrow w = \sum_{i=1}^n \alpha_i y_i x_i \tag{1.12}$$

$$\frac{L(w, \alpha, \xi, \beta, b)}{\partial b} = 0 \Longrightarrow \sum_{i=1}^n \alpha_i y_i = 0$$

$$\frac{L(w, \alpha, \xi, \beta, b)}{\partial \xi_i} = 0 \Longrightarrow C - \beta_i - \alpha_i \tag{1.13}$$

$$\begin{aligned} \alpha_i(y_i(w^T x_i + b) - 1 + \xi) &= 0 \\ \beta_i \xi_i &= 0 \\ \xi_i &\geq 0, \; \beta_i \geq 0, \alpha_i \geq 0 \\ \beta_i + \alpha_i &= C \end{aligned}$$

En remplaçant cette équation dans la fonction objective 1.11, on obtient le problème dual suivant :

$$\left\{ \begin{array}{c} \max L(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \sum_{i=1}^n \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \quad i = 1...n \end{array} \right. . \tag{1.14}$$

La seule différence avec les SVM à marge dure est que les α_i ne peuvent pas dépasser C, ils peuvent être dans l'un des trois cas suivants :

$\alpha_i = 0 \Longrightarrow \beta_i = C \Longrightarrow \xi_i = 0 : x_i$ est bien classé,

$0 < \alpha_i < C \Longrightarrow \beta_i > 0 \Longrightarrow \xi_i = 0 \Longrightarrow y_i(w^T x_i + b) = 1 : x_i$ est un vecteur support et est appelé dans ce cas vecteur support non borné (unbounded),

$\alpha_i = C \Longrightarrow \beta_i = 0 \Longrightarrow \xi_i \geq 0 \Longrightarrow y_i(w^T x_i + b) = 1 - \xi_i$: xi est un vecteur support appelé dans ce cas vecteur support borné (bounded). Si $0 < \xi_i < 1$, x_i est bien classé, sinon x_i est mal classé. Ces conditions sur les α_i sont appelées les conditions de Karush-Kuhn-Tucker (KKT), elles sont très utilisées par les algorithmes d'optimisation pour rechercher les α_i optimums et par conséquent l'hyperplan optimal.

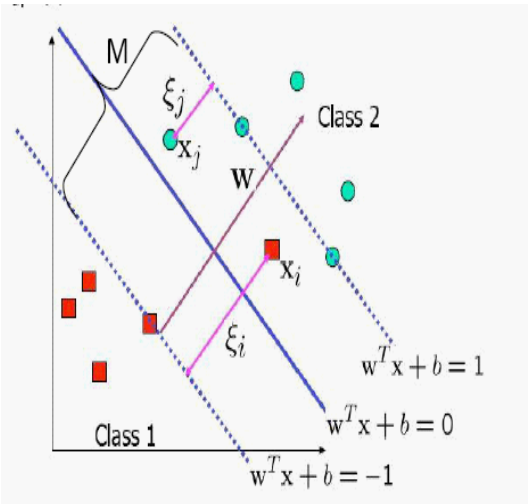


*Figure*6 :Marge souple.

1.2.3 Les noyaux

Jusqu'à présent, les machines à vecteurs de support permettent de trouver une règle pour classer les données lorsque celles-ci sont linéairement séparables. Cependant, il existe bien des cas pour lesquels il est impossible de séparer entièrement les données avec un hyperplan. Telles qu'elles ont été présentées jusqu'à présent, les SVM sont incapables de traiter un tel problème, puisqu'il est alors impossible que les contraintes

$$y_i(w^T x_i + b) \geq 1$$

soient toutes respectées.

La détermination d'une telle fonction non linéaire est très difficile voire impossible. Pour cela les données sont amenées dans un espace où cette fonction devient linéaire, L'espace où se trouvent les données avant d'être transformées est appelé l'espace d'entrée (input space),cette astuce permet de garder les mêmes modèles de problèmes d'optimisation vus dans les sections précédentes, utilisant les SVMs basées essentiellement sur le principe de séparation linéaire. Cette transformation d'espace est réalisée souvent à l'aide d'une fonction $\phi : \mathbb{R}^n \rightarrow H$ appelé "Feature mapping" et le nouvel espace H est appelé espace des caractéristiques "Features space". Dans ce nouvel espace des caractéristiques, la fonction objective à optimiser devient :

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j) \rangle \tag{1.15}$$

Où $\langle \phi(x_i), \phi(x_j) \rangle$ est le produit scalaire des deux images des vecteurs x_i et $x_j$dans le nouvel espace H et dont le résultat est un scalaire. Dans le calcul de l'optimum de la fonction, on utilise une astuce appelée "Noyau" ("Kernel"). Toutefois, l'utilisation des transformations pose certains problèmes. En e et, outre le fait qu'il faille choisir une bonne transformation, il faut l'appliquer à toutes les données, puis effectuer les calculs avec ces données transformées, c'est- à-dire dans l'espace de redescription .C'est ici que la formulation duale du problème d'optimisation prend toute son importance. En effet, on remarque que lorsque le problème est sous sa forme duale, les données de l'ensemble d'apprentissage n'apparaissent que dans un produit scalaire avec d'autres données du même ensemble. Ceci amène à définir comme suit une fonction appelée noyau (kernel) :

$$\begin{array}{rcl} K & : & \mathbb{R}^n \ \ \mathbb{R}^n \longrightarrow \mathbb{R} \\ (x_i, x_j) & \longrightarrow & \langle \phi(x_i), \phi(x_j) \rangle \end{array}$$

Cette fonction prend en entrée deux points dans l'espace d'entrée et calcule leur produit scalaire dans l'espace caractéristique . L'avantage d'une telle fonction est qu'il n'est pas nécessaire de chercher une transformation aux données. Ce calcul peut se faire directement à partir des données de l'espace d'entrée. Grâce au concept de noyau, il est possible de réécrire le problème dual de cette manière :

$$\begin{cases} \max L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \\ \alpha_i \geq 0 \ i = 1...n \end{cases}$$

On remarque que de cette manière, lorsque la fonction noyau est connue, la transformation $\phi(x)$ n'apparaît nulle part, ni dans le problème, ni dans l'application de la solution. Par conséquent, Cette fonction noyau permet,donc de faire tous les calculs nécessaires sans avoir à se préoccuper de la dimension de l'espace des caractéristiques H.

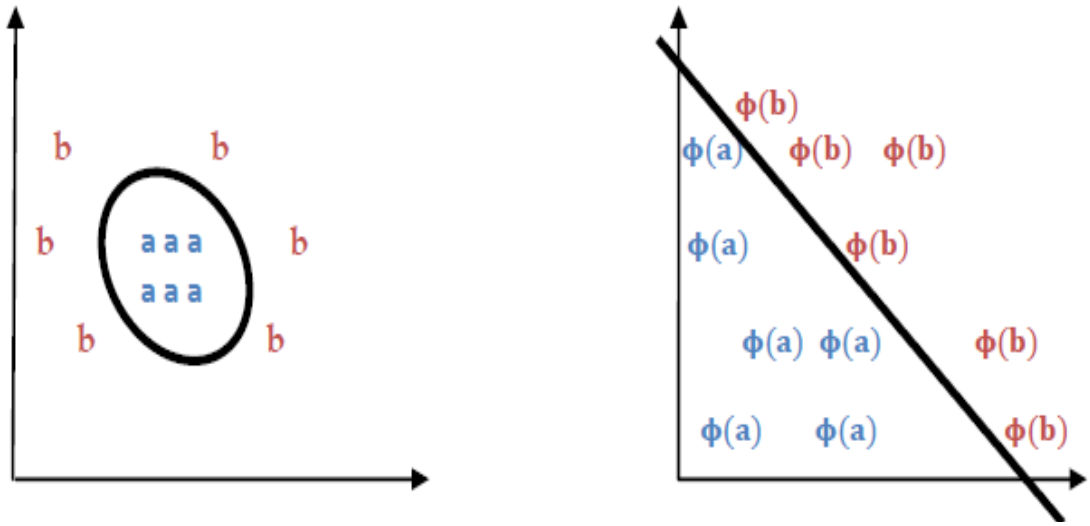


Figure 7 : Espace de projection des donnes non linairement sparable

Exemples de noyaux

– Noyau linéaire : Si les données sont linéairement séparables, on n'a pas besoin de changer d'espace, et le produit scalaire suffit pour définir la fonction de décision :

$$k(x_i, x_j) = x_i^T x_j.$$

– Noyau polynomial : Le noyau polynomial élève le produit scalaire à une puissance naturelle d :

$$k(x_i, x_j) = (x_i^T x_j)^d.$$

Si $d = 1$le noyau devient linéaire. Le noyau polynomial dit non homogène $k(x_i, x_j) = (x_i^T x_j + C)^d$ est aussi utilisé.

– Noyau RBF : Les noyaux RBF (Radial Basis functions) sont des noyaux qui peuvent être écrits sous la forme :$k(x_i, x_j) = f(d(x_i x_j))$où d est une métrique sur X et f est une fonction dans $\mathbb{R}$. Un exemple des noyaux RBF est le noyau Gaussien :

$$k(x_i, x_j) = \exp(\frac{-\|x_i - x_j\|^2}{2\sigma^2}),$$

où σ est un réel positif qui représente la largeur de bande du noyau.

Chapitre 2

La méthode Active-set

Le corps d'une méthode SVM est le problème dual quadratique donné au chapitre précédent. La résolution efficace de ce programme déterminera une bonne fonction classificatrice. Dans ce chapitre nous présenterons une des méthodes les plus utilisées pour résoudre des problèmes quadratiques convexes, à savoir La méthode Active-set. En général, un programme quadratique est formulé comme suit

$$\begin{cases} \min \ q(x) = \frac{1}{2}x^T G x^T + x^T c \\ \quad a_i^T x = b_i \quad i \in J \\ \quad a_i^T x \geq b_i \quad i \in I \end{cases}, \tag{2.1}$$

où la matrice $G \in \mathbb{R}^{nxn}$ est symétrique semi défini positive, c, $a_i \in \mathbb{R}^{nx1}$ et I,J des ensembles d'indices fini. Par conséquent, une telle méthode est appropriée pour résoudre le problème quadratique associé au SVMs. Si on connait à l'avance l'ensemble des contraintes actives à la solution optimale $A\,(x^*)$, la résolution du problème 2.1 est équivalente à la résolution du problème suivant :

$$\begin{cases} \min \ q(x) = \frac{1}{2}x^T G x^T + x^T c \\ \quad a_i^T x = b_i \quad i \in A\,(x^*) \end{cases}$$

Cependant, l'ensemble $A(x^*)$ n'est pas toujours connu à l'avance, l'idée de la méthode active set est de construire itération après itération l'ensemble $A(x^*)$ à partir d'un l'ensemble $A(x^0)$ pour un certain point initial x^0. Nous donnons dans la section suivante une description de la méthode pour une étude approfondie le lecteur peut consulté l'ouvrage [Numerical Optimization].

2.1 Description de la méthode

Etant à l'itération k et disposant d'une solution admissible x_k , nous définissons l'ensemble W_k appelé **l'ensemble de travail** (working set)

$$W_k = \left\{ i \in A(x^{*)} \ \text{ telle que } \ \{a_i\}_{i \in A(x^*)} \quad \text{son linéairement indépendants} \right\}.$$

Nous vérifiant d'abord si le point x_k minimise la fonction objectif q sur W_k .Sinon, nous cherchons la direction p solution du problème restreint qu'on va définir ci-dessous. Pour se faire, écrivons la direction désirée comme suit :

$$p = x - x_k \Longrightarrow x = x_k + p. \tag{2.2}$$

En remplaçant x dans la fonction objectif $q(x)$ du problème donné en 2.1, on trouve

$$\begin{aligned} q(x) &= q(x_k + p) = \frac{1}{2}(x_k + p)^T G(x_k + p) + (x_k + p)^T c \\ &= \frac{1}{2}p^T Gp + p^T Gx_k + x_k^T c + p^T c + \frac{1}{2}x_k Gx_k \\ &= \frac{1}{2}p^T Gp + p^T g_k + \rho_k. \end{aligned} \tag{2.3}$$

Avec $g_k = Gx_k + c$ et $\rho_k = \frac{1}{2}x_k Gx_k + x_k^T c$, puisque ρ_k est indépendant de p ,la résolution du problème

$$\begin{cases} \min \frac{1}{2}p^T Gp + p^T g_k + \rho_k \\ a_i^T p = 0, i \in W_k \end{cases}$$

est équivalente au problème appelé problème restreint

$$(\mathcal{P}_k) \begin{cases} \min \frac{1}{2}p^T Gp + p^T g_k \\ a_i^T p = 0, i \in W_k \end{cases} \tag{2.4}$$

Notons p_k la solution optimale de $\mathcal{P}_k$, et construisons le nouveau itéré $x^{k+1} = x_k + \alpha p_k$, la présence du $\alpha \geq 0$, est pour assurée l'admissibilité de x^{k+1}.Plus précisément, pour $i \in W_k$, $a_i^T x^{k+1} = a_i^T(x^k + \alpha p_k) = a_i^T x^k + \alpha a_i^T p_k = b_i$.On a deux cas possible pour p_k, soit $p_k = 0$ ou $p_k \neq 0$. Supposons que la solution de 2.4 est $p_k \neq 0$. Si $x_k + p_k$ vérifie toutes les contraintes, alors $x^{k+1} = x_k + \alpha^k p_k$. Sinon on cherche un paramètre $\alpha_k \in [0, 1]$, telle que $x^{k+1} = x_k + \alpha^k p_k$ soit admissible pour le problème **??** . On peut donner une définition explicite pour α_k, en étudiant le comportement des contraintes. Quand $i \in W_k$ les contraintes seront satisfaites, en effet, $a_i^T(x_k + \alpha p_k) = a_i^T x_k + \alpha^k a_i^T p_k \geq b_i$, $\forall\ \alpha \geq 0$.Si par contre $i \notin W_k$ on a deux cas :

Si $a_i^T p_k \geq 0$ on a

$$\begin{aligned} a_i^T(x_k + \alpha p_k) &= a_i^T x_k + \alpha^k a_i^T p_k \\ &\geq a_i^T x_k \\ &\geq b_{i.} \end{aligned}$$

Si $a_i^T p_k < 0$ on aura

$$\begin{aligned} a_i^T(x_k + \alpha p_k) \quad &\geq \quad b_i \Longleftrightarrow a_i^T x_k + \alpha^k a_i^T p_k \geq b_i \\ &\Longleftrightarrow \quad \alpha^k a_i^T p_k \geq b_i - a_i^T x_k, \end{aligned}$$

si et seulement si

$$\alpha^k \leq \frac{b_i - a_i^T x_k}{a_i^T p_k}$$

on définie α^k comme suite :

$$\alpha^k = \min\{1, \min_{\substack{i \notin w_k \\ et\ a_i^T p_k < 0}} (\frac{b_i - a_i^T x_k}{a_i^T p_k})\}.$$

S'il existerai $i_0 \notin w_k$ tel que

$$0 \leq \alpha^k = \frac{b_{i_0} - a_{i_0}^T x_k}{a_{i_0}^T p_k} \leq 1.$$

Alors, la contrainte d'indice i_0 est appelée **contrainte bloquante**. On a trois cas de figure peuvent se présenter pour α^k :

- Soit on a une contrainte bloqante i_0, $0 \leq \alpha^k \leq 1$, par conséquent l'ensemble de travail sera mise à jour comme suit $W_{k+1} = W_k \cup \{$ i_0 l'indice de la contrainte bloquante$\}$.

- $\alpha^k = 1$, mais pas issue d'une contrainte bloquante,

- $\alpha^k = 0$, donc une contrainte blanquante a été trouvé, mais qui n'a pas encore introduite dans l'ensemble de travail W_k.

On continue à itérer jusqu'à recontrer un point $\hat{x}$ qui minimise la fonction objectif du problème 2.4, sous les contraintes $a_i^T x_k = b_i$ $\forall i \in \hat{W}$, on reconnait ce point quant la solution du problème restaint est $p^* = 0$. Les conditions d'optimalité de Lagrange donnent

$$\nabla_p L(p^*, \lambda) = g^k - \sum_{i \in \hat{w}} \hat{\lambda}_i a_i = 0, \tag{2.5}$$

avec les $\{a_i\}_{i \in w_k}$ sont linéairement indépendants .On trouve les $\hat{\lambda}$ en résolevant le systeme 2.5, on aura deux cas :

- si $\forall i \in \hat{W}$ $\hat{\lambda}_i \geq 0$ stop $x^* = \hat{x}$

sinon $\exists j \in \hat{W}$ et $\hat{\lambda}_i < 0$, on peut considérer $\hat{\lambda}_{j_0} = \arg\min \left\{ \hat{\lambda}_j \backslash \hat{\lambda}_j < 0 \right\}$, ensuite mettre à jour l'ensemble de travail par : $W_{k+1} = W_k / \{j_0\}$.

2.2 Algorithme Active-Set

Initialisation : calculer un point initial x_0. Trouver w_0 l'ensemble de travail initial.
Pour $k = 1, 2, ...$Résoudre le problème restreint 2.4.
Si $p_k = 0$
calculer $\hat{\lambda}_i$ vérifiant 2.5 , avec $\hat{W} = W_k$.
Si

$$\hat{\lambda}_i \geq 0, \forall i \in \hat{W} \cap I,$$

Stop la solution est $x^* = \hat{x}$;
Sinon, trouver

$$j = \arg \min_{i \in \hat{W} \cap I} \left\{ \hat{\lambda}_i \ \ / \hat{\lambda}_i < 0 \right\}.$$

Mettre à jour

$$W_{k+1} = W_k / \{j\}$$

et

$$x^{k+1} = x^k.$$

Fin de si
Sinon ($p_k \neq 0$)
calculer

$$\alpha^k = \min\{1, \min_{\substack{i \notin w_k \\ et\ a_i^T p_k < 0}} (\frac{b_i - a_i^T x_k}{a_i^T p_k})\}.$$

$$x^{k+1} = x_k + \alpha^k p_k.$$

S'il y à une contrainte bloquante ,on obtient

$$W_{k+1} = W_k \cup \{i_0 \text{l'indice de la contrainte bloquante}\}.$$

Sinon

$$W_{k+1} = W_k$$

Fin de si

$$k = k + 1$$

Fin de pour.

2.2.1 Exemple

Considéront le problème

$$\begin{cases} \min q(x) = (x_1 - 1)^2 + (x_2 - 2.5)^2 \\ x_1 - 2x_2 + 2 \geq 0 \\ -x_1 - 2x_2 + 6 \geq 0 \\ -x_1 + 2x_2 + 2 \geq 0 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{cases}.$$

Appliquons la méthode Active-Set, avec $x^0 = (2, 0)^T$. Déterminons les données

$$G = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, c = \begin{bmatrix} -2 \\ -5 \end{bmatrix}, cst = 29/4$$

$$A = \begin{bmatrix} 1 & -2 \\ -1 & -2 \\ -1 & 2 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, b = \begin{bmatrix} -2 \\ -6 \\ -2 \\ 0 \\ 0 \end{bmatrix}$$

calculons $g_0 = \nabla q(x_0) = (2, -5)^T$ et considérons le sous problème

$$(\mathcal{P}_0) \begin{cases} \min \frac{1}{2} p^T G p + p^T g_0 \\ a_i^T p = 0 \ i \in w_0 \end{cases}$$

On doit résoudre le système

$$\begin{bmatrix} G & -A_{w_0}^T \\ A_{w_0} & 0 \end{bmatrix} \begin{bmatrix} p \\ \lambda \end{bmatrix} = \begin{bmatrix} -2 \\ 5 \\ 0 \\ 0 \end{bmatrix}$$

le système est bien

$$\begin{bmatrix} 2 & 0 & 1 & 0 \\ 0 & 2 & -2 & -1 \\ -1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \widehat{\lambda}_3 \\ \widehat{\lambda}_5 \end{bmatrix} = \begin{bmatrix} -2 \\ 5 \\ 0 \\ 0 \end{bmatrix}$$

On trouve que $p = 0$
On doit trouver $\widehat{\lambda}_3$ et $\widehat{\lambda}_5$ en résolvant le système

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix} \widehat{\lambda}_3 + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \widehat{\lambda}_5 = \begin{bmatrix} 2 \\ -5 \end{bmatrix}$$

On trouve que

$$\left(\widehat{\lambda}_3, \widehat{\lambda}_5\right) = (-2, -1)$$

Mise à jour de W

$$W_1 = \{5\} \text{ et } x_1 = x_0$$

Comme $x_1 = x_0$, $g_1 = g_0$
On doit résoudre le problème

$$(\mathcal{P}_1) \begin{cases} \min \frac{1}{2} p^T G p + p^T g_0 \\ a_i^T p = 0 \ i \in w_1 \end{cases}$$

Le sous problème $(\mathcal{P}_1)$ donne la résolution

$$\begin{bmatrix} G & -A_{w_1}^T \\ A_{w_1} & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} -2 \\ 5 \\ 0 \end{bmatrix}$$

Explicitement on a le système

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} -2 \\ 5 \\ 0 \end{bmatrix}$$

On trouve que $p_1 = (-1, 0)^T \neq 0$
Clculons le pas α_1
Déterminons les $i : i \notin W_1$ et $a_i^T p_1 \prec 0$, On trouve que $\left\{a_1^T p_1 \prec 0 \text{ et } a_4^T p_1 \prec 0\right\}$ et

$$\alpha_1 = \min\left\{1, \frac{b_1 - a_1^T x_1}{a_1^T p_1}, \frac{b_4 - a_4^T x_1}{a_4^T p_1}\right\} = \min\{1, 4, 4\} = 1$$

Il n'y a pas de contraintes bloquantes et le nouveau iétré

$$x_2 = x_1 + \alpha_1 p_1 = (1, 0)^T \text{ et } W_2 = W_1$$

Calculons $g_2 = Gx_2 + c = (0, -5)^T$
On résout le sous problème et on trouve que $p_2 = (0, 0)^T$ et $\widehat{\lambda}_5 = -5$
et donc

$$W_3 = \varnothing \text{ et } x_3 = x_2.$$

On continue d'itérer jusqu'à trouver la solution optimale.

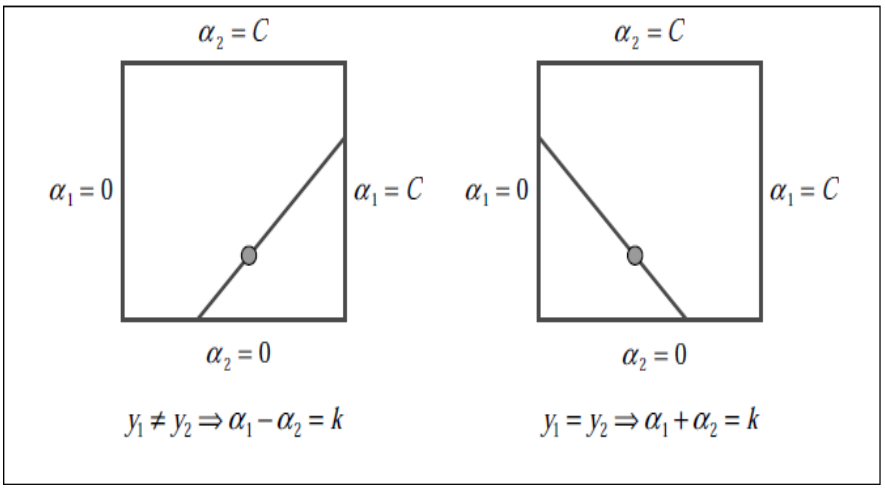
Chapitre 3

Optimisation Minimal Séquentielle

3.1 Introduction

Du à sa grande taille, le problème quadratique qui apparait dans les SVMs (SVMQP), ne peut être résolu par une méthode quadratique standard, comme celle introduite au chapitre précédent. On a vue que cette méthode nécessite la résolution d'un programme quadratique avec contraintes égalités à chaque itération, et qui nécessite l'inversion de la matrice Hessienne à chaque étape, ceci peut être trop couteux en mémoire et temps d'exécution. Dans ce chapitre, nous présenterons un algorithme simple qui peut résoudre rapidement le problème SVMQP sans aucun stockage supplémentaire de matrice et sans utiliser un solveur d'optimisation quadratique à chaque étape. La méthode est appelée Optimisation Minimal Séquentielle (En anglais : Sequential Minimal Optimization (SMO)). La méthode SMO décompose le problème QP global en sous-problèmes QP, mais contrairement à la méthode précédente, SMO choisit de résoudre le plus petit problème d'optimisation possible à chaque étape. Le problème d'optimisation le plus petit nécessite que deux multiplicateurs de Lagrange, car les multiplicateurs de Lagrange doivent obéir à une contrainte d'égalité linéaire. À chaque étape, SMO choisit deux multiplicateurs de Lagrange pour les optimiser conjointement, trouve la valeur optimale associée à ces deux multiplicateurs et met à jour le SVM afin de refléter les nouvelles valeurs optimales (voir figure 2). L'avantage de SMO réside dans le fait que la résolution de deux multiplicateurs de Lagrange peut être effectuée analytiquement. Ainsi, la boucle interne de l'algorithme peut être exprimée en une simple exécution, plutôt que d'invoquer un solver de programmation quadratique. Il est claire qu'un nombre important de sous-problèmes d'optimisation seraient résolus au cours de l'algorithme, mais chaque sous-problème est si rapide à résoudre que le problème global est résolu rapidement. En outre, SMO ne nécessite aucun stockage en mémoire supplémentaire au stockage d'une matrice 2×2. Ainsi, des problèmes SVMs de grand taille peuvent être résolu dans un ordinateur personnel. Chaque itération de SMO contient trois étapes :

- Une méthode analytique pour la résolution des deux Multiplicateurs de Lagrange.
- Une heuristique pour choisir les deux multiplicateurs à optimiser.
- Une méthode pour le calcul de b.



3.2 La méthode Analytique

En chaque étape SMO optimise deux multiplicateurs de Lagrange. Sans restreindre la généralité, supposons que ces deux multiplicateurs sont α_1 et α_2.La fonction objectif donnée en 1.14 devient :

$$\left\{ W(\alpha_1, \alpha_2) = \alpha_1 + \alpha_2 - \frac{1}{2}K_{11}\alpha_1^2 - \frac{1}{2}K_{22}\alpha_2^2 - sK_{12}\alpha_1\alpha_2 - y_1\alpha_1 v_1 - y_2\alpha_2 v_2 + W_{const} \right. \quad (3.1)$$

où

$$K_{ij} = K(x_i)\, K(x_j)$$

$$v_i = \sum_{j=3}^{N} y_j \alpha_j^{old} K_{ij} = f^{old}(x_i) + b^{old} - y_1\alpha_1^{old}K_{1i} - y_2\alpha_2^{old}K_{2i}, \quad (3.2)$$

et la variable avec un indice "old" indique la valeur à l'itération précédente. W_{const} est le terme qui ne dépend pas de α_1 ou α_2 et $s = y_1 y_2$. A chaque étape on doit trouver le maximum le long du segment défini par la contrainte égalité dans 1.14. cette contrainte égalité peut être exprimer par

$$\alpha_1 + s\alpha_2 = \alpha_1^{old} + s\alpha_2^{old} = \gamma.$$

La fonction objectif le long de la contrainte égalité peut être exprimer uniquement en fonction de α_2 :

$$W = \gamma - s\alpha_2 + \alpha_2 - \frac{1}{2}k_{11}(\gamma - s\alpha_2)^2 - \frac{1}{2}k_{22}\alpha_2^2 - \frac{1}{2}sk_{12}(\gamma - s\alpha_2)\alpha_2 + y_1(\gamma - s\alpha_2)v_1 - y_2\alpha_2 v_2 + W_{const} \quad (3.3)$$

Le point stationnaire de la fonction objective peut être déterminer par :

$$\frac{dW}{d\alpha_2} = sk_{11}(\gamma - s\alpha_2) - k_{22}\alpha_2 + k_{12}\alpha_2 - sk_{12}(\gamma - s\alpha_2) + y_1 v_1 - s - y_2 v_2 + 1 = 0. \quad (3.4)$$

Si la dérivée seconde, le long de la contrainte égalité linéaire est negative, alors le maximum de la fonction objective peut être exprimé par :

$$\alpha_2^{new}(k_{11} + k_{22} - 2k_{12}) = s(k_{11} - k_{12})\gamma + y_2(v_1 - v_2) + 1 - s. \quad (3.5)$$

Développer l'équation pour γ et v le rendement :

$$\alpha_2^{new}(k_{11}+k_{22}-2k_{12}) = \alpha_2^{old}(k_{11}+k_{22}-2k_{12}) + y_2 f(x_1) - f(x_2) + y_2 - y_1) \tag{3.6}$$

Les extrémités du segment de ligne diagonal peuvent être exprimées tout simplement. Sans perte de généralité, l'algorithme calcule d'abord le deuxième multiplicateur de Lagrange α_2 et calcule les extrémités du segment de ligne diagonal en termes de α_2. Si la cible y_1 n'est pas égale à la cible y_2, les limites suivantes s'appliquent à α_2 :

$$L = \max(0, \alpha_2^{old} - \alpha_1^{old}) \qquad H = \min(C, C + \alpha_2^{old} - \alpha_1^{old}). \tag{3.7}$$

Si la cible y_1 est égale à la cible y_2 , les limites suivantes s'appliquent à α_2 :

$$L = \max(0, \alpha_1^{old} + \alpha_2^{old} - C) \qquad H = \min(C, \alpha_2^{old} - \alpha_1^{old}). \tag{3.8}$$

La deuxième dérivée de la fonction objective selon la ligne diagonale peut s'exprimer comme suit :

$$\eta = 2K(x_1, x_2) - K(x_1, x_1) - K(x_2, x_2). \tag{3.9}$$

La prochaine étape du SMO consiste à calculer l'emplacement du maximum de la fonction d'objectif dans l'équation (1.14) tout en permettant aux deux multiplicateurs de Lagrange de changer. Dans des circonstances normales, η est inférieur à zéro. Dans ce cas, SMO calcule le maximum le long de la direction de la contrainte :

$$\alpha_2^{new} = \alpha_2^{old} + \frac{y_2(E_1 - E_2)}{\eta}. \tag{3.10}$$

où $E_i = f^{old}(x_i) - y_i$, est l'erreur sur le ième exemple d'apprentissage. Ensuite, le minimum restreint est trouvé en projetant le minimum non contraint aux extrémités du segment de ligne :

$$\alpha_2^{new\ clipped} = \begin{cases} H & si \quad \alpha^{new} \geq H \\ \alpha_2^{new} & si \quad L < \alpha_2^{new} < H \\ L & si \quad \alpha_2^{new} \leq L. \end{cases} \tag{3.11}$$

La valeur de α_1 est calculé à partir du nouveau α_2 projeté:

$$\alpha_1^{new} = \alpha_1 + s(\alpha_2 - \alpha_2^{new\ clipped}). \tag{3.12}$$

3.3 Heuristiques pour le choix des deux multiplicateurs

SMO optimise toujours et modifie deux multiplicateurs de Lagrange à chaque étape, les deux multiplicateurs de Lagrange ayant précédemment violés les conditions de KKT dans l'itération précédente. C'est-à-dire, SMO modifiera toujours deux multiplicateurs de Lagrange, pour se déplacer en montée dans la fonction d'objectif projetée dans l'unidimensionnel sous-espace admissible .SMO maintiendra également toujours des multiplicateurs de Lagrange admissibles. Par conséquent, la fonction d'objectif global augmentera à chaque étape et l'algorithme convergera asymptotiquement (mettre la référence de Osuna). Afin d'accélérer la convergence, SMO utilise des heuristiques pour choisir les multiplicateurs de

Lagrange à optimiser conjointement. Il existe deux types d'heuristiques : une pour le premier multiplicateur de Lagrange et une pour le second. Le choix de la première heuristique fournit la boucle externe de l'algorithme SMO. La boucle externe se déroule tout au long de l'ensemble de l'apprentissage, en déterminant les exemples violant les conditions de KKT 1.12. Le SVM est ensuite mis à jour à l'aide de ces deux nouvelles valeurs, et la boucle externe reprend la recherche des violateurs des conditions de KKT. Afin d'accélérer le processus, la boucle externe ne se déroule pas toujours à travers l'ensemble total d'apprentissage. Après une seule étape entière, la boucle extérieure ne se déroule que sur les exemles dont. Si un exemple viole les conditions de KKT, il est alors candidat pour une optimisation immédiate. Une fois un tel candidat est trouvé, le second multiplicateur est choisi celon la deuxième heuristique et les deux multiplicateurs sont optimisés conjointement. L'admissibilité par rapport au problème dual (1.14) est toujours maintenueles multiplicateurs de Lagrange ne sont ni 0 ni C (les exemples vérifiant $0 < \alpha < C$ appelés les exemples non bornés). La boucle externe effectue des étapes répétées sur les exemples non bornés, jusqu'à ce que tous les exemples non bornés obéissent aux conditions de KKT avec une précision d'ordre ϵ. La boucle externe se rétablit à nouveau sur l'ensemble total d'apprentissage. La boucle externe continue d'alterner entre les étapes simples sur l'ensemble d'apprentissage total et les étapes multiples sur le sous-ensemble nonbornés jusqu'à ce que l'ensemble d'apprentissage complète obéit aux conditions KKT avec une précision d'ordre ϵ. En ce point, l'algorithme s'arrête.

Une fois qu'un premier multiplicateur de Lagrange est choisi, SMO choisit le deuxième multiplicateur de Lagrange pour maximiser l'optimisation conjoint. L'évaluation de la fonction noyau prend beaucoup de temps, donc SMO se rapproche de la taille de l'étape par la valeur absolue du numérateur dans l'équation 3.11 ($|E_1 - E_2|$). Si E_1 est positif, SMO choisit un exemple avec une erreur minimale E_2. Si E_1 est négatif, SMO choisit un exemple avec une erreur maximale E_2.

3.4 Le calcul de b

La résolution (1.14) pour les multiplicateurs de Lagrange α ne détermine pas la valeur de b du SVM, donc b doit être calculé séparément. Après chaque étape, b est recalculé, de sorte que les conditions KKT seront vérifiés pour les deux exemples optimisés. La valeur suivante de b_1 est valide lorsque le nouvel α_1 n'est pas au bornes, parceque sa force la sortie du SVM à être soit y_1 lorsque l'entrée est x_1 :

$$b_1 = E_1 + y_1(\alpha_1^{new} - \alpha_1^{old})K(x_1, x_1) + y_2(\alpha_2^{new\ clipped} - \alpha_2^{old})K(x_1, x_2) + b^{old} \tag{3.13}$$

Le suivant b_2 est valide lorsque le nouveau α_2 n'est pas au bornes, car il force l'outputof du SVM à y_2 lorsque l'entrée est x_2 :

$$b_2 = E_2 + y_1(\alpha_1^{new} - \alpha_1^{old})K(x_1, x_2) + y_2(\alpha_2^{new\ clipped} - \alpha_2^{old})K(x_2, x_2) + b^{old}. \tag{3.14}$$

Quand deux $b_1$et b_2 sont valides, ils sont forcément égaux. Lorsque les deux nouveaux multiplicateurs de Lagrange sont au bornes et si L n'est pas égal à H, alors l'intervalle entre b_1 et b_2 correspond à tous les seuils comsistent avec les conditions KKT. SMO choisit le seuil à mi-chemin entre b_1 et b_2.

Chapitre 4

Experimentations numériques

L'objectif de ce chapitre est de faire une comparaison en temps d'exécution, entre la méthode Optimisation Séquentielle Minimal (SMO) et Active Set sur différentes base de données. L'étude sera faite avec le solver *svmtrain* de Matlab de la boite à outil statistique (Statitics Toolbox), dont la syntaxe est comme suit

$$SVMStruct = svmtrain(Training, Group, Name, Value),$$

où Training est la matrice des données d'apprentissage et Group est le vecteur des classe correspondante à chaque ligne de la matrice Training. *svmtrain* donne une certaine liberté de choix de paramètre tel que le noyau utilisé, la constante C, dans notre étude nous avons opté pour le choix de la méthode par, $Name = method$ et $Value = QP$ ou SMO. La méthode QP n'est rien d'autre que la méthode active-set, quand la boite à outil Optimisation (Optimization Toolbox) est disponible, alors que SMO est la méthode de Platt exposé au chapitre précédent.. la comparaison est faite sur les cinq bases de donnée suivantes de l'UCI Machine Learning Repository [ ].

- diabetes : Cette base est constituée de 768 exemples d'apprentissage . Chaque exemple décrit par 8 attributs (caratiristique), le $9^{ième}$ représente les classes.
- heart_scale : Cette base est constituée de 270 exemples d'apprentissage . Chaque exemple décrit par 12 attributs (caratiristique), le $13^{ième}$ représente les classes.
- bupa : Cette base est constituée de 345 exemples d'apprentissage . Chaque exemple décrit par 6 attributs (caratiristique), le$7^{ième}$ représente les classes.
- liver-disorders : Cette base est constituée de345 exemples d'apprentissage .Chaque exemple décrit par 6 attributs (caratiristique), le$7^{ième}$ représente les classes.
- spectf : Cette base est constituée de 80 exemples d'apprentissage .Chaque exemple décrit par 44 attributs (caratiristique), le $1^{ième}$ représente les classes.

	diabetes	heart_scale	bupa	liver-disorders	spectf
Temps d'exécution avec Active-Set	68.1195	5.4320	2.8154	2.9101	0.8733
Temps d'exécution avec SMO	2.0649	1.4118	1.4384	1.4204	2.135

*Table*1.*Résultats de comparaison entre SMO et Active Set.*

Chapitre 5

Conclusion

Dans ce mémoire, on a présenté d'une manière simple l'approche introduit par Vladimir Vapnik pour l'apprentissage, à savoir les « Support Vecteur Machine ». On a donné une vision générale et une vision purement mathématiques des SVM. Cette méthode de classification est basée sur la recherche d'un hyperplan qui permet de séparer au mieux des ensembles de données. On a exposé le cas linéairement séparable et le cas non linéairement séparables qui nécessitent l'utilisation de 'astuce noyau (Kernel). Cette méthode est applicable pour des taches de classification à deux classes, mais il existe des extensions pour la classification multi classes. Nous avons présenté les aspects liés à la mise en œuvre d'un algorithme performent, appelé Optimisation Séquentielle minimal (SMO) qui peut résoudre des problèmes SVMs de grand taille. Aussi, une étude comparative a été présentée avec une méthode classique de programmation quadratique (QP) à savoir la méthode Active-Set afin de justifier notre choix. Comme d'autres algorithmes d'apprentissage, SMO décompose un gros problème QP en une série de problèmes QP plus petits. Contrairement à d'autres algorithmes, SMO utilise les problèmes QP les plus petits possibles, qui sont résolus rapidement et analytiquement, ce qui améliore considérablement son temps de calcul et le stockage en mémoire.

Bibliographie

[1] D. Francoeur, Machine a Vecteur de Support -Une Introduction, CaMUS .1, 2010, page : 7-25.

[2] J. Nocedal, S. J. Wright, Numerical Optimization, Second Edition, Springer, 2006.

[3] J. C. Platt, Fast Training of Support Vector Machines Using Sequential Minimal Optimisation.

[4] H. Mohamadally, SVM : Machines à Vecteurs de Support ou Séparateurs à Vastes Marges .16 janvier 2006.

[5] M. Georgiopoulo, .Practical Implementation of the Active Methode for Support Vector Machine Training whith Semi -Definite Kernels.

[6] Michael C. Ferris Olvi L. Mangasarian Stephen J. Wright .Linear Programming with MATLAB ,University of Wisconsin–Madison
Madison, Wisconsin,2007.

[7] S. Khellat- Kihel, .Les séparateurs a vaste marge Bi-classes.Mémoire de Master, Université des Sciences et de la Technologie d'Oran Usto M.B.2011-2012.

[8] Sequential Minimal Optimisation for SVM.

[9] Machines à vecteur support, http ://abdelhamid-djeffal.net/web_documents/courssvm.pdf.