ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE DE LA PRODUCTION AUTOMATISÉE
M.Ing.

PAR
MATHIAS MAHOUSONZOU ADANKON

OPTIMISATION DE RESSOURCES POUR LA SÉLECTION DE MODELE DES SVM

MONTRÉAL, LE 16 SEPTEMBRE 2005

CE MÉMOIRE A ÉTÉ ÉVALUÉ
PAR UN JURY COMPOSÉ DE :

M. Mohamed Cheriet, directeur de mémoire
Département de génie de la production automatisée à l'École de technologie supérieure

M. Richard Lepage, président du jury
Département de génie de la production automatisée à l'École de technologie supérieure

M. Alain Biem, examinateur externe
IBM at Watson Research Center, N.Y, USA

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC
LE 5 AOÛT 2005
À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

OPTIMISATION DE RESSOURCES POUR LA SÉLECTION DE MODELE DES SVM

Mathias Mahouzonsou Adankon

SOMMAIRE

La sélection de modèle, optimisation des hyper-paramètres, est une étape très importante pour garantir une forte performance aux SVM. Cette sélection est souvent réalisée par la minimisation d'un estimé de l'erreur en généralisation basé sur les bornes du "leave-one-out" comme le "radius-margin bound" et sur certaines mesures de performance comme GACV(Generalized Approximate Cross Validation), l'erreur empirique, etc. Ces méthodes de sélection de modèle automatique nécessitent l'inversion de la matrice de Gram-Schmidt ou la résolution d'un problème d'optimisation quadratique supplémentaire, ce qui est très coûteux en temps de calcul et en mémoire lorsque la taille de l'ensemble d'apprentissage devient importante.

Dans ce mémoire, nous proposons une méthode rapide basée sur une approximation du gradient de l'erreur empirique avec une technique d'apprentissage incrémental; ce qui réduit les ressources requises en termes de temps de calcul et d'espace mémoire. Avec l'approximation du gradient, nous n'avons pas besoin d'inverser la matrice de Gram-Schmidt avant de calculer le gradient de l'erreur empirique. L'apprentissage incrémental, quant à lui, permet d'optimiser de façon parallèle les paramètres et les hyper-paramètres de la machine, afin de réduire le temps de calcul. Notre méthode testée sur des bases de données synthétiques et réelles a produit des résultats probants confirmant notre approche. En outre, nous avons noté que le gain de temps s'accroît lorsque la taille de l'ensemble d'apprentissage devient large, ce qui rend notre méthode intéressante dans le cas des applications réelles.

Nous avons aussi développé une nouvelle expression pour les SVM avec la formulation de la marge molle «soft margin» L1, ce qui permet d'inclure l'hyper-paramètre C dans les paramètres du noyau. Ainsi, nous pouvons résoudre le problème de la différentiation de C et dans certains cas réduire le nombre des hyper-paramètres dans la sélection de modèle.

OPTIMIZING RESOURCES IN MODEL SELECTION FOR SVM

Mathias Mahouzonsou Adankon

ABSTRACT

Tuning SVM hyperparameters is an important step for achieving a high-performance learning machine. This is usually done by minimizing an estimate of generalization error based on the bounds of the leave-one-out (loo) as radius-margin bound and on the performance measure as GACV, empirical error, etc. These usual automatic methods used to tune the hyperparameters require an inversion of the Gram-Schmidt matrix or a resolution of an extra quadratic programming problem. In the case of a large dataset these methods require the addition of huge amounts of memory and a long CPU time to the already significant resources used in the SVM training.

In this dissertation, we propose a fast method based on an approximation of the gradient of the empirical error along with incremental learning, which reduces the resources required both in terms of processing time and of storage space. With the gradient approximation, we do not need to invert the Gram-Schmidt matrix for computing the gradient of the empirical error. The incremental learning makes it possible to optimize both the parameters of the SVM and the hyperparameters and to drastically save on computing time. We tested our method on many benchmarks which have produced promising results confirming our approach. Furthermore, we notice that the gain time increases when the dataset is large.

We also develop a new expression for SVM with L1 soft-margin formulation that makes it possible to include the hyper-parameter C in the kernel parameters. Then, we can resolve the problem of C differentiation and in certain cases reduce the number of the hyperparameters for model selection.

REMERCIEMENTS

J'adresse toute ma reconnaissance au PCBF-ACDI, pour le soutien financier, sans lequel ce travail n'aurait pu être réalisé.

Je tiens à remercier mon directeur de recherche, M. Mohamed Cheriet, professeur à l'École de Technologie Supérieure, pour son suivi, pour son soutien et pour ses vifs encouragements tout au long de ce mémoire de maîtrise.

Je remercie également M. Alain Biem, chercheur Senior chez IBM à Watson pour s'être intéressé à mon travail et avoir accepté de faire partie de mon jury. Aussi, ma gratitude à M. Richard Lepage, professeur à l'École de Technologie Supérieure, qui a accepté de présider ce jury.

Un grand merci à tous les membres du Laboratoire d'Imagerie, de la Vision et de l'Intelligence Artificielle pour leur sympathie et leur gentillesse.

Enfin, je dédie ce travail à Pascaline et à toute ma famille pour leur soutien et leur encouragement. Que ces lignes leur témoignent toute ma reconnaissance.

TABLE DES MATIÈRES

Page

SOMMAIRE i
ABSTRACT ii
REMERCIEMENTS iii
TABLE DES MATIÈRES iv
LISTE DES TABLEAUX vii
LISTE DES FIGURES viii
LISTE DES ABRÉVIATIONS ET DES SIGLES x
INTRODUCTION 1
CHAPITRE 1 CLASSIFIEURS À NOYAUX ET SVM 5
1.1 Introduction 5
1.2 Méthodes des noyaux 6
1.2.1 Principe et Théorème de Mercer 6
1.2.2 Propriétés des noyaux 7
1.2.3 Exemples des noyaux 8
1.3 Description de certains algorithmes utilisant la méthode des noyaux 11
1.3.1 Kernel PCA 11
1.3.2 Kernel Fisher discriminant 13
1.3.3 L'Analyse discriminante généralisée (GDA) 14
1.3.4 Feature Vectors Selection (FVS) 17
1.3.5 Relevance Vector Machine (RVM) 19
1.4 Principe et Modélisation des SVM 21
1.4.1 Risque structurel et dimension VC 21
1.4.2 Principe des SVM : maximisation de la marge de séparation 24
1.4.3 SVM linéaire et séparable 26
1.4.4 SVM linéaire et marge molle (cas non séparable) 28
1.4.5 SVM non linéaire : "astuce du noyau" («kernel trick») 31
1.5 Algorithmes d'apprentissage des SVM 32
1.5.1 Méthode de décomposition 32
1.5.2 Algorithme de Joachims : méthode de décomposition améliorée 34
1.5.3 Optimisation séquentielle minimale : SMO 35
1.6 Classification multi-classe avec les SVM 37
1.6.1 Approche un-contre-tous 37
1.6.2 Approche un-contre-un 38
1.7 Conclusion 39

CHAPITRE 2 SÉLECTION DE MODÈLE POUR LES SVM: ÉTAT DE L'ART.....40
2.1 Introduction.....40
2.2 Technique de la validation croisée.....40
2.2.1 Théorie de la validation croisée.....40
2.2.2 Validation croisée généralisée approchée (GACV).....41
2.3 Bornes de "leave-one-out" pour les SVM.....42
2.3.1 Nombre de vecteurs de support.....42
2.3.2 Borne de Jaakkola-Haussler.....43
2.3.3 Borne de Opper-Winter.....43
2.3.4 Borne de Joachims.....44
2.3.5 Borne basée sur l'écartement des vecteurs de support.....44
2.3.6 Borne "Rayon-Marge".....45
2.4 Erreur empirique.....46
2.5 Conclusion.....50

CHAPITRE 3 STRATÉGIES D'OPTIMISATION DES RESSOURCES.....51
3.1 Introduction.....51
3.2 Approximation du gradient de l'erreur empirique.....51
3.3 Optimisation des paramètres avec l'apprentissage incrémental.....61
3.3.1 Rappel des techniques d'apprentissage incrémental.....61
3.3.2 Jumelage de l'apprentissage incrémental avec la sélection de modèle.....63
3.4 Analyse de l'espace mémoire et de la complexité.....68
3.4.1 Espace mémoire.....69
3.4.2 Coût de calcul.....69
3.5 Conclusion.....75

CHAPITRE 4 NOUVELLE FORMULATION DU SVM-L1.....76
4.1 Introduction.....76
4.2 Description de la nouvelle formulation.....76
4.3 Équation de l'hyperplan de séparation avec la nouvelle formulation.....79
4.4 Propriétés de $\tilde{k}(x_i.x_j) = Ck(x_i.x_j)$.....80
4.5 Avantages de cette nouvelle formulation pour la sélection de modèle.....83
4.6 Conclusion.....83

CHAPITRE 5 EXPÉRIMENTATIONS ET DISCUSSIONS.....85
5.1 Introduction.....85
5.2 Performance de nos stratégies dans la sélection de modèle.....85
5.2.1 Données artificielles.....85
5.2.2 Benchmark UCI.....86
5.2.3 Base de données MNIST.....88
5.3 Analyse des propriétés de nos stratégies.....90
5.3.1 Impact de l'approximation du gradient.....90
5.3.2 Impact de l'apprentissage incrémental.....92
5.3.3 Impact global des deux stratégies en fonction de la taille.....93
5.3.4 Impact de la taille initiale de S.....95
5.4 Expérimentation de la nouvelle formulation du SVM-L1.....96

5.5 Conclusion 99

CONCLUSION GÉNÉRALE 100

ANNEXES :

1. Détails du calcul du coût d'apprentissage 102
2. Calcul détaillé du coût d'estimation du gradient approché 104
3. Calcul détaillé du coût d'estimation du gradient total 106
4. Comparaison des taux d'erreur par problème biclasse entre la nouvelle et l'ancienne formulation 108

BIBLIOGRAPHIE 110

LISTE DES TABLEAUX

Page

Tableau I	Fonctions de couplage	39
Tableau II	Description des bases provenant de UCI utilisées	86
Tableau III	Taux d'erreur en test trouvés par différents algorithmes de sélection de modèles	87
Tableau IV	Résultats obtenus avec MNIST en utilisant les différents Couplages	89
Tableau V	Résultats obtenus avec la nouvelle formulation sur les cinq bases de UCI en comparaison avec les résultats testés en section 5.2.2	97
Tableau VI	Tableau donnant le nombre d'éléments mal classés en test (biclasse) par la nouvelle formulation vs. par l'ancienne formulation	98

LISTE DES FIGURES

Page

Figure 1 Représentation d'un exemple de problème bi-classe.....5

Figure 2 Résultat de la projection des données de l'exemple de la fig.1.....6

Figure 3 Variation du noyau polynomial.....9

Figure 4 Variation du noyau gaussien.....10

Figure 5 Phénomènes de sous-apprentissage et de sur-apprentissage.....22

Figure 6 Dimension VC de l'ensemble des fonctions linéaires de R2.....23

Figure 7 Représentation de l'hyperplan et des vecteurs de support.....25

Figure 8 Algorithme de décomposition pour l'apprentissage des SVMs (extraite de [24]).....34

Figure 9 Variation de la probabilité estimée en fonction de la sortie du SVM.....48

Figure 10 Sélection de modèle basée sur l'erreur empirique.....49

Figure 11 Représentation des données du problème XOR.....55

Figure 12 Variation de l'erreur empirique E et de l'erreur de test en validation au cours de l'optimisation des paramètres du noyau avec les données synthétiques du problème XOR représentées sur la figure 11-a.....57

Figure 13 Variation de l'erreur empirique E et de l'erreur de test en validation au cours de l'optimisation des paramètres du noyau avec les données synthétiques du problème XOR représentées sur la figure 11-b.....58

Figure 14 Variation de l'erreur empirique E et de l'erreur de test en validation au cours de l'optimisation des paramètres du noyau avec les données de la base UCI désignée par Thyroïde.....60

Figure 15 Variation du taux de bonne classification sur l'ensemble de test en fonction de la taille de l'ensemble d'apprentissage.....64

Figure 16 Algorithme d'apprentissage jumelé avec l'optimisation du noyau.....65

Figure 17 Schéma synoptique du jumelage de l'apprentissage incrémental avec la sélection de modèle.....66

Figure 18 Comportement de la taille de ΔS en fonction de la norme du gradient.....67

Figure 19 Comparaison entre le temps de calcul requis en fonction de la taille de l'ensemble d'apprentissage (Gradient total vs Gradient approximé).....90

Figure 20 Variation du taux de réduction du temps de calcul en fonction de la taille de l'ensemble d'apprentissage (Gradient total vs Gradient approximé).....91

Figure 21 Comparaison entre le temps de calcul requis en fonction de la taille de l'ensemble d'apprentissage (Apprentissage incrémental vs Apprentissage normal).....92

Figure 22 Variation du taux de réduction du temps de calcul en fonction de la taille de l'ensemble d'apprentissage (Apprentissage incrémental vs Apprentissage normal).....93

Figure 23 Comparaison entre le temps de calcul requis en fonction de la taille de l'ensemble d'apprentissage.....94

Figure 24 Variation du taux de réduction du temps de calcul en fonction de la taille de l'ensemble d'apprentissage.....94

Figure 25 Courbes montrant l'impact de la taille initiale de S.....95

LISTE DES ABRÉVIATIONS ET DES SIGLES

SVM	Machine à Vecteur de Support
RBF	Noyau à base radiale
KMOD	Kernel with Moderate Decreasing
MLP	Perceptron multi-couche
KPCA	Analyse en Composantes Principales non linéaire
KFD	Discriminant de Fisher non linéaire
LDA	Analyse discriminante linéaire
GDA	Analyse discriminante non linéaire
FVS	Vecteurs sélectionnés dans l'espace augmenté
RVM	Relevance vector machine
γ	paramètre du noyau RBF, inverse de la variance
VC	Vapnik-Chernovenkis
h	Dimension VC
$R(\alpha)$	Risque réel
R_{emp}	Risque empirique
ρ	Marge d'un hyperplan
R^d	Espace de réels de dimension d
x_i	vecteur d'entrée d'indice i
y_i	étiquette du vecteur d'entrée d'indice i
ℓ	nombre d'exemples d'apprentissage
N	nombre d'exemples de validation
w	vecteur orthogonal à l'hyperplan optimal
b	paramètre de biais de l'hyperplan optimal
α	ensemble des multiplicateurs de lagrange, paramètres du SVM
C	paramètre de compromis dans le SVM
ξ_i	variable ressort associé à l'observation x_i

$L(w,b,\alpha)$	lagrangien primaire du SVM
$W(\alpha)$	lagrangien dual du SVM
N_{VS}	nombre de vecteurs de support
$sign(x)$	fonction signe, retournant le signe de l'argument x
SMO	optimisation par minimisation séquentielle
LOO	Leave-One-Out
GACV	Erreur de validation croisée généralisée
QP	Quadratique Problem
KKT	Karush-Kun-Tucker
D	diamètre de la boule englobant les points d'apprentissage
R	rayon de la boule englobant les points d'apprentissage
E_i	Erreur empirique associée à l'observation x_i
$\hat{p}_i$	Probabilité a posteriori de l'observation x_i
θ	vecteur des hyper-paramètres
H	matrice de Gram-schmidt modifiée
UCI	University of California, Irvine
MNIST	Modified NIST database
PCBF	Programme Canadien de Bourses de la Francophonie
ACDI	Agence Canadienne de Développement International

INTRODUCTION

La reconnaissance de formes dont le but consiste à associer une étiquette (une classe) à une donnée qui peut se présenter sous forme d'une image ou d'un signal, est un axe fort important du domaine de l'intelligence artificielle, qui trouve application dans beaucoup de systèmes comme les interfaces visuelles, l'analyse de données, la bioinformatique, le multimédia, etc. Plusieurs méthodes ont été développées dans ce domaine, en particulier les réseaux de neurones avec le perceptron multicouche (MLP). Les réseaux de neurones ont fait l'objet de nombreuses recherches et ont donné des résultats impressionnants comme moyen de prédiction. Et depuis longtemps, les perceptrons multicouches ont été utilisés avec succès pour résoudre de nombreux problèmes de classification. Mais, de plus en plus, avec la diversité des applications de reconnaissance de formes, plusieurs problèmes non linéaires faisant intervenir la classification sont rendus complexes. Ainsi, depuis quelques années, pour contourner la complexité des fonctions de décision construites pour les problèmes non linéaires, les recherches ont donné naissance à de nouvelles méthodes de classification basées sur les noyaux (*kernel methods*) qui s'avèrent plus robustes et plus simples que d'autres méthodes telles que les couches cachées introduites dans les réseaux de neurones.

La méthode des noyaux est une technique récente en reconnaissance de formes. Elle consiste à projeter les données qui sont initialement non séparables dans un autre espace de dimension plus élevée où elles peuvent le devenir. Les classifieurs utilisant la méthode des noyaux, contrairement aux classifieurs traditionnels, construisent la fonction de décision dans un nouvel espace autre que l'espace des caractéristiques d'entrée; ce qui permet de réduire la complexité de la fonction de décision tout en gardant une bonne performance du classifieur. Les machines à vecteurs de support, issues des travaux de Vapnik, utilisent cette technique des noyaux qui permet de traiter linéairement dans le nouvel espace les problèmes préalablement non linéaires.

Les machines à vecteurs de support constituent une famille de classifieurs basés sur le principe du risque structurel qui leur confère un fort pouvoir de généralisation. La minimisation du risque structurel permet d'éviter le phénomène de sur-apprentissage des données à la convergence du processus d'apprentissage des machines de classification. Ainsi, le risque structurel permet de garantir une estimation moins biaisée du risque réel. Cependant, le choix des hyper-paramètres (les paramètres du noyaux et la valeur de pénalisation des variables d'écarts) définissant l'architecture d'une SVM affecte de façon significative sa performance. Alors, il faut utiliser une bonne stratégie de sélection pour optimiser les valeurs des hyper-paramètres d'une SVM afin d'espérer une bonne performance.

Plusieurs travaux de sélection de modèle pour les machines à vecteurs de support ont été effectués. En 2001, Chapelle et al. [1] ont proposé pour la première fois une méthode automatique pour sélectionner les hyper-paramètres d'un SVM en se basant sur des bornes de l'erreur en généralisation dérivée de LOO (Leave-one-out). Il s'agit de la dimension VC donnant la borne «rayon-marge» (radius-margin) et celle mesurant l'écartement des vecteurs de support (span bound). Mais ces méthodes développées pour réaliser l'ajustement automatique des paramètres nécessitent l'inversion de la matrice de Gram-Schmidt des vecteurs de support et la résolution d'un problème d'optimisation quadratique additionnel, ce qui requiert un temps de calcul important et un espace mémoire non négligeable. En 2003, Kai-Min et al. [4] utilisent le même critère "radius-margin" pour optimiser automatiquement les hyper-paramètres sans inverser la matrice de Gram- Schmidt dans le calcul du gradient, mais ils ne pouvaient pas se passer de la résolution du problème QP additionnel.

Récemment, un nouveau critère de sélection de modèle pour les SVM appelé l'erreur empirique a été développé par Ayat et al. [5, 6]. Cette méthode minimise l'erreur de généralisation à travers un ensemble de validation. Ce critère est une fonction linéaire simple qui ne nécessite guère la résolution d'un autre problème QP, à part celui de

l'apprentissage du SVM. De plus, nous avons la dérivabilité de la fonction coût qui permet de quantifier l'erreur empirique. Cependant, l'apprentissage des machines à vecteurs de support demande d'importantes ressources en temps et en stockage au fur et à mesure que la base d'apprentissage devient large. Alors, cette procédure de sélection de modèle automatique, malgré qu'elle soit plus rapide et simple en complexité que les autres critères, reste coûteuse en temps de calcul et en espace mémoire pour de grandes bases de données.

Dans le cadre de ce mémoire de maîtrise, nous nous sommes donnés pour objectif l'optimisation des ressources au cours de la procédure de sélection de modèle en utilisant l'erreur empirique compte tenue des vertus de ce critère. L'optimisation des ressources vise à réduire le temps de calcul du CPU et l'espace de stockage mémoire afin de faciliter l'intégration de la sélection de modèle dans des applications réelles avec moins de coût. Ainsi, pour arriver à nos fins, nous proposons :

- une approximation du gradient de l'erreur empirique, ce qui nous permet de déterminer le gradient sans inverser la matrice de Gram-Schmidt, réduisant ainsi la complexité du calcul du gradient.

- une stratégie d'apprentissage incrémental des SVM, technique qui permet de faire évoluer parallèlement la procédure d'optimisation des paramètres du modèle et l'apprentissage dans une technique de jumelage.

Par ailleurs, nous nous sommes intéressés à l'optimisation du paramètre C au même titre que les autres hyper-paramètres.

Le présent mémoire est organisé comme suit :

Le chapitre 1 présente le concept et les classifieurs utilisant la technique des noyaux et particulièrement les machines à vecteurs de support.

Le chapitre 2 est consacré à l'état de l'art sur la sélection de modèle pour les machines à vecteurs de support.

Le chapitre 3 décrit les différentes stratégies que nous avons développées pour l'optimisation des ressources dans la sélection de modèle des machines à vecteurs de support.

En chapitre 4, nous présentons une nouvelle formulation du SVM-L1, ce qui permet d'inclure le paramètre C dans le choix des paramètres du noyau pour la sélection de modèle.

Le chapitre 5 présente les expériences effectuées dans le cadre de ce projet, les résultats obtenus et les analyses relatives.

Enfin, nous terminons ce rapport par une conclusion générale.

CHAPITRE 1

CLASSIFIEURS À NOYAUX ET SVM

1.1 Introduction

Pour éviter la complexité de la fonction de décision au cours de l'apprentissage, il est souvent intéressant de projeter les données dans un autre espace de caractéristiques de dimension plus élevée. Dans ce nouvel espace de représentation, les données qui étaient difficilement séparables peuvent le devenir avec une fonction de décision moins complexe. Comme exemple, considérons deux classes de données représentées par deux cercles concentriques de rayon respectif 1 et 0,5, voir figure 5 [7].

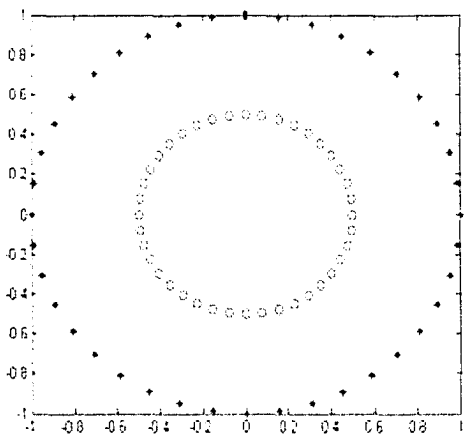


Figure 1 Représentation d'un exemple de problème bi-classe

Les données sont représentées par des vecteurs en dimension 2, réalisons une projection de ces données en dimension 3 par l'application définie ci-après :

$$\begin{aligned} \phi: \quad & R^2 \rightarrow R^3 \\ & (x_1, x_2) \mapsto (x_1^2, x_2^2, \sqrt{2}x_1x_2) \end{aligned} \tag{1.1}$$

Le résultat de cette transformation appliquée à toutes les données représentées sur la figure 1 est donné en figure 2. Nous pouvons noter que pour séparer les deux classes de données dans ce nouvel espace il suffit d'utiliser une fonction linéaire (un plan) adéquate, ce qui n'était pas possible dans l'espace initial de dimension 2. La précédente technique utilisée pour obtenir une séparation linéaire est l'idée principale de la méthode des noyaux.

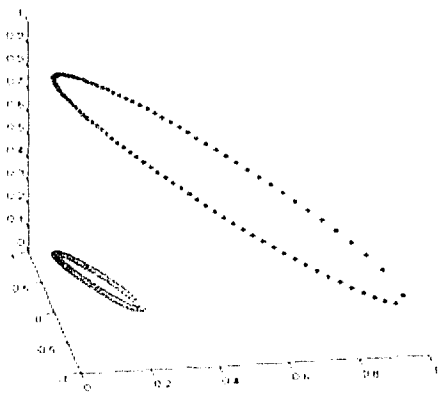


Figure 2 Résultat de la projection des données de l'exemple de la figure 1

1.2 Méthodes des noyaux

1.2.1 Principe et Théorème de Mercer

La méthode des noyaux [8, 9] consiste à projeter les données de l'espace R^d dans un autre espace de dimension plus élevée où les données qui étaient non linéairement séparables peuvent le devenir. Désignons par F le nouvel espace, la fonction de projection $\phi : R^d \rightarrow F$ est une transformation non linéaire, qui est évaluée implicitement. En réalité, on définit une fonction $k : R^d \times R^d \rightarrow R$ avec

$k(x, y) = \phi(x).\phi(y)$ pour réaliser la projection. La fonction k ainsi définie est appelée *noyau* et elle représente le produit scalaire dans l'espace F.

En considérant l'exemple de la section 1.1, la fonction noyau vaut $k(x, y) = (x_1^2, x_2^2, \sqrt{2}x_1x_2).(y_1^2, y_2^2, \sqrt{2}y_1y_2) = (x_1y_1 + x_2y_2)^2 = (x.y)^2$.

Pour définir l'existence d'une fonction noyau, on se sert souvent du théorème de Mercer [10] qui s'énonce comme suit :

Soit une fonction symétrique $k : \mathrm{X} \times \mathrm{X} \rightarrow R$ avec $\mathrm{X} \subset R^d$, il existe une fonction ϕ telle que $k(x, y) = \phi(x).\phi(y)$ si et seulement si pour toute fonction f élément de l'ensemble des fonctions définies sur X, telle que :

$$\int_X f(x)^2 dx \text{ existe}$$

on a

$$\int_{X \times X} k(x, y) f(x) f(y) dx dy \geq 0$$

La positivité de l'intégrale permet de définir l'existence du noyau, qui est un produit scalaire dans un espace de Hilbert.

1.2.2 Propriétés des noyaux

Commutativité

Soient x et y deux éléments de l'espace $\mathrm{X} \subset R^d$, la fonction noyau étant symétrique, nous avons la commutativité :

$$k(x, y) = k(y, x)$$

Inégalité de Cauchy-Schwartz

Considérons toujours x et y deux éléments de $X \subset R^d$, nous avons :

$$[k(x,y)]^2 = [\phi(x).\phi(y)]^2 \leq \|\phi(x)\|^2 . \|\phi(y)\|^2$$

Par conséquent :

$$[k(x,y)]^2 \leq k(x,x).k(y,y)$$

Combinaison de noyaux

Soient k_1 et k_2 deux noyaux définis de $X \times X$, $X \subset R^d$, dans R. Les fonctions suivantes sont aussi des noyaux :

1. $k(x,y) = k_1(x,y) + k_2(x,y)$
2. $k(x,y) = ak_1(x,y)$ avec $a \in R^+$
3. $k(x,y) = k_1(x,y).k_2(x,y)$
4. $k(x,y) = \exp(k_1(x,y))$
5. $k(x,y) = p(k_1(x,y))$ avec $p : R \rightarrow R$ une fonction polynomiale à coefficients positifs.

1.2.3 Exemples de noyaux

Noyau polynomial

La fonction noyau polynomial de degré n est définie par :

$$k(x,y) = (ax.y + b)^n \text{ avec } (a,b) \in R^2$$

Le noyau polynomial exprimant un produit scalaire dépend de la direction des vecteurs et de leur norme. Ainsi, l'image d'un vecteur x avec ceux de même direction produit de

fortes valeurs de noyau. En figure 3, nous montrons la variation du noyau $k(x, y) = (x.y + 1)^3$ en fonction de différents vecteurs $y \in [0,20] \times [0,20]$ associés à $x = (10,10)$.

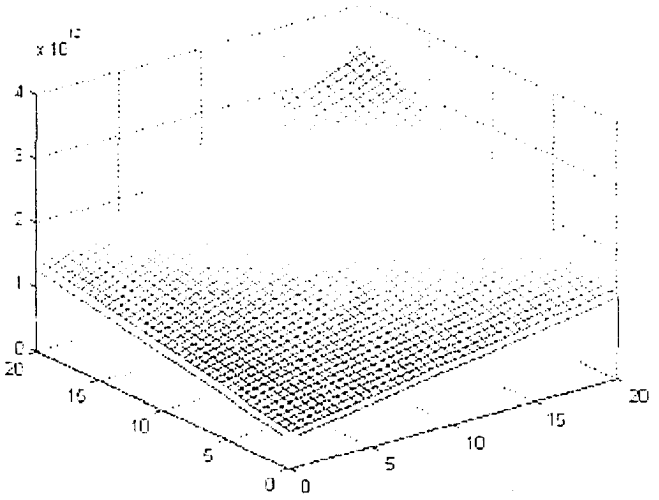


Figure 3 Variation du noyau polynomial

La dimension du nouvel espace *F*, lorsqu'on utilise le noyau polynomial de degré *n* pour projeter les données de l'espace *X* vers *F*, vaut :

$$\dim F = \frac{(n + \dim X - 1)!}{n!(\dim X - 1)!}$$

Noyau gaussien

Le noyau gaussien est dérivé de la fonction RBF. Il dépend de la distance euclidienne entre les deux vecteurs dans l'espace de départ. Il est défini par l'expression ci-après :

$$k(x, y) = \exp(-\frac{\|x - y\|^2}{\sigma^2}) \text{ avec } \sigma \in R$$

Par cette expression, nous notons que la valeur du noyau, contrairement au noyau polynomial, est indépendante de la direction. La figure 4 montre la variation du noyau gaussien avec $\sigma = 3$ en dimension 2, pour différents vecteurs $y \in [0,20]\times[0,20]$ associés à $x = (10,10)$.

Le noyau gaussien est le plus utilisé dans les applications. La dimension de F est infinie pour ce noyau.

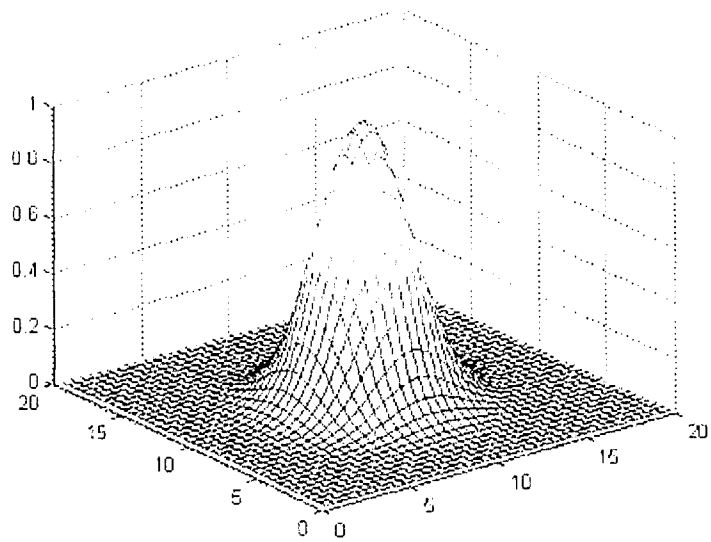


Figure 4 Variation du noyau gaussien

Autres noyaux

Nous pouvons aussi citer comme fonctions noyaux :

- le noyau linéaire pour caractériser l'absence de projection dans un autre espace

$$k(x, y) = x.y$$

- le noyau Laplacien

$$k(x, y) = \exp(-\frac{\|x - y\|}{c})$$

- le noyau sigmoïdal qui ne vérifie pas le théorème de Mercer

$$k(x, y) = \tanh(ax.y + b)$$

- le noyau inverse multiquadratique

$$k(x, y) = (\|x - y\| + c^2)^{-1/2}$$

- le noyau KMOD [11]

$$k(x, y) = a\left(\exp(\frac{\gamma^2}{\|x - y\|^2 + \sigma^2}) - 1 \right)$$

1.3 Description de certains algorithmes utilisant la méthode des noyaux

1.3.1 Kernel PCA

Le KPCA [12, 13] est la généralisation de l'Analyse en Composantes Principales. L'idée de base est la réduction de la dimension de l'espace des caractéristiques en ne retenant que les composantes de forte variance qualifiées de principales.

Soit un ensemble d'observations centrées $x_i \in R^d$ des vecteurs lignes avec $i = 1, ..., \ell$; les composantes principales sont déterminées par les valeurs propres $\lambda > 0$ et les vecteurs propres v non nuls de l'équation (1.2).

$$\lambda v = Cv \tag{1.2}$$

où C désigne la matrice de covariance des observations :

$$C = \frac{1}{\ell}\sum_{i=1}^{\ell} x_i x_i^T \tag{1.3}$$

Ainsi :

$$\lambda v = Cv = \frac{1}{\ell}\sum_{i=1}^{\ell} (x_i.v)x_i^T \tag{1.4}$$

Et on en déduit alors que tous les vecteurs v avec $\lambda \neq 0$ sont des éléments du sous-espace générés par les observations x_i.

En considérant l'espace augmenté pour le KPCA, la relation 1.4 s'écrit :

$$\lambda v = Cv = \frac{1}{\ell}\sum_{i=1}^{\ell} (\phi(x_i).v)\phi(x_i) \tag{1.5}$$

Nous avons aussi la même conclusion, les vecteurs v avec $\lambda \neq 0$ sont des combinaisons linéaires des éléments de l'espace augmenté.

$$v = \sum_{j=1}^{\ell} \alpha_j \phi(x_j) \tag{1.6}$$

En multipliant l'équation 1.5 par $\phi(x_s)$ avec $s = 1,...,\ell$ et en la combinant avec l'équation 1.6, nous avons :

$$\lambda\sum_{j=1}^{\ell} \alpha_j \left(\phi(x_s).\phi(x_j)\right) = \frac{1}{\ell}\sum_{j=1}^{\ell} \alpha_j \left[\phi(x_s).\left(\sum_{i=1}^{\ell} \phi(x_i)\left(\phi(x_i).\phi(x_j)\right)\right)\right] \tag{1.7}$$

pour tout $s = 1,...,\ell$.

Si on désigne par K, la matrice carrée formée par le noyau des observations, les ℓ équations obtenues de 1.7 s'écrivent de façon matricielle :

$$\ell\lambda K\alpha = K^2\alpha \tag{1.8}$$

Les solutions de l' équation précédente sont les mêmes que celles de l'équation ci-après :

$$\ell\lambda\alpha = K\alpha \tag{1.9}$$

L'analyse en composantes principales dans l'espace augmenté permet ainsi de déterminer les vecteurs v de projection de l'équation 1.6 en résolvant l'équation 1.9. Et pour une observation x donnée, la projection suivant une composante principale v^k dans l'espace F est déterminée par :

$$v^k\Phi(x) = \sum_{j=1}^{\ell} \alpha_j^k(\Phi(x_j).\Phi(x)) = \sum_{j=1}^{\ell} \alpha_j^k k(x_j, x) \tag{1.10}$$

1.3.2 Kernel Fisher discriminant

L'idée de base de "Kernel Fisher discriminant" [14, 15] est de résoudre le problème de la discrimination linéaire suivant la théorie de Fisher dans l'espace augmenté F.

En effet le principe de Fisher est de trouver la direction de projection des données pour une meilleure séparabilité des classes. Les critères utilisés sont :

- maximiser la distance entre les moyennes des classes
- minimiser la variance à l'intérieur des classes.

Le but revient alors à déterminer les vecteurs ω de projection qui maximisent le coefficient de Rayleigh :

$$J(\omega) = \frac{\omega^t S_B \omega}{\omega^t S_W \omega} \tag{1.11}$$

où $S_B = (m_2 - m_1)(m_2 - m_1)^t$ et $S_W = \sum_{k=1,2} \sum_{i=1}^{n_l} (x_i - m_k)(x_i - m_k)^t$

avec m_k la moyenne et n_k le nombre d'observations dans une classe.

Par similarité, la relation (1.11) devient (1.12) dans l'espace F en remplaçant les x_i par $\phi(x_i)$ et ω par la combinaison des éléments de F, c'est-à-dire $\omega = \sum_{i=1}^{l} \alpha_i \phi(x_i)$.

$$J(\alpha) = \frac{\alpha^t M \alpha}{\alpha^t N \alpha} \tag{1.12}$$

où

$$M = (m_2 - m_1)(m_2 - m_1)^t \text{ et } N = \sum_{k=1,2} K_k (I - U_k) K_k^t$$

avec $m_k = \frac{1}{n_k} K_k.(1,1,...,1)^t$ et U_k = matrice carré $n_k \times n_k$ d'éléments égaux à $1/n_k$.

Les vecteurs α qui maximisent J(α) sont les vecteurs propres de la matrice $N^{-1}M$ qui ont de fortes valeurs propres associées.

La projection des données se fait par la relation :

$$\omega.\Phi(x) = \sum_{i=1}^{l} \alpha_i (\Phi(x_i).\Phi(x)) = \sum_{i=1}^{l} \alpha_i k(x_i, x) \tag{1.13}$$

Le KFD est utilisé pour les problèmes de deux classes. Pour plusieurs classes, on utilise l'analyse discriminante généralisée.

1.3.3 L'Analyse discriminante généralisée (GDA)

L'analyse discriminante linéaire (LDA) est une technique statistique. Cette méthode basée sur la minimisation de la variance intra-classe par rapport à la variance inter-classe

a eu beaucoup de succès dans les problèmes de classification. Mais en ce qui concerne les problèmes non linéaires, la LDA n'est pas efficace. Cette limite[1] de la LDA fut franchie par l'approche des noyaux, puisque dans le « feature espace »[2] les données qui n'étaient pas linéairement séparables peuvent le devenir. Cette nouvelle approche de discrimination a été développée par Baudat et al. dans [16].

L'analyse discriminante linéaire part de la connaissance de la partition en classes des individus d'une population et cherche les combinaisons linéaires des variables décrivant les individus, qui conduisent à la meilleure discrimination entre les classes. L'idée de base est de créer une méthode pour choisir parmi les combinaisons linéaires des variables celle qui maximise l'homogénéité de chaque classe. Une fois cette combinaison choisie, on procède à la projection des données sur les axes qui sont les plus discriminants suivant la variance inter-classe.

Soit un problème de c classes, et désignons par n_k le nombre d'observations contenues dans la classe $k=1,\ldots,c$ avec $\ell = \sum_{k=1}^{c} n_k$ le nombre total d'observations.

Dans l'espace augmenté F, avec des données centrées, la matrice de covariance est notée par :

$$C = \frac{1}{\ell}\sum_{i=1}^{\ell}\phi(x_i)\phi(x_i)^T = \frac{1}{\ell}\sum_{k=1}^{c}\sum_{j=1}^{n_k}\phi(x_{kj})\phi(x_{kj})^T \qquad (1.14)$$

où x_{kj} désigne le *jème* observation de la *kème* classe.

Notons B la matrice de covariance des moyennes des classes dans l'espace F, représentant l'inertie inter-classe.

---

[1] Les limitations de la LDA ont été aussi résolus par les MLP

[2] «feature space» désigne l'espace augmenté F

$$B = \frac{1}{\ell} \sum_{k=1}^{c} n_k \bar{\phi}_k \bar{\phi}_k^T \tag{1.15}$$

où $\bar{\phi}_k = \frac{1}{n_k} \sum_{j=1}^{n_k} \phi(x_{kj})$ représente la valeur moyenne de la *kème* classe.

La combinaison linéaire discriminante, qui permet de minimiser les variances intra-classe tout en maximisant les variances inter-classes, est constituée des vecteurs propres v solutions de l'équation :

$$\lambda C v = B v \tag{1.16}$$

Les fortes valeurs propres de l'équation 1.16 sont celles qui maximisent l'équation 1.17 :

$$\lambda = \frac{v^T B v}{v^T V v} \tag{1.17}$$

Puisque les vecteurs propres v sont des combinaisons linéaires des éléments de F, il existe des coefficients α_{pq} *(p=1,...,c et q=1,...,n_p)* tels que

$$v = \sum_{p=1}^{c} \sum_{q=1}^{n_p} \alpha_{pq} \phi(x_{pq}) \tag{1.18}$$

Ce qui permet d'en déduire que l'équation 1.17 est équivalente à l'égalité ci-après :

$$\lambda = \frac{\alpha^t K W K \alpha}{\alpha^t K K \alpha} \tag{1.19}$$

où $W = (W_k)_{k=1,...,c}$ est une matrice bloc diagonale avec W_k une matrice $n_k \times n_k$ dont tous les termes sont égaux à $1/n_k$.

Connaissant les valeurs de α vérifiant 1.19, une donnée de test z est projetée en utilisant l'expression suivante :

$$v.\phi(z) = \sum_{p=1}^{c} \sum_{q=1}^{n_p} \alpha_{pq} k(x_{pq}, z) \tag{1.20}$$

Une approche du GDA séquentielle a été développée par Fahed et al.[17] dans le but de contourner les problèmes de calcul matriciel survenus lorsque la taille de l'ensemble d'apprentissage devient très importante.

1.3.4 Feature Vectors Selection (FVS)

La méthode des FVS développée par Baudat et Anouar dans [7] et [18] consiste à sélectionner des vecteurs de l'espace des caractéristiques F qui seront les représentants de tous les autres vecteurs. En réalité, on essaie de créer un sous-espace représentatif de F. Les vecteurs formant ce sous-espace noté *S* sont sélectionnés à partir d'une approche géométrique et ils permettent de capturer la structure géométrique des données d'apprentissage dans l'espace F.

Soit L le nombre de vecteurs sélectionnés $X_S = \{x_{S1}, x_{S2}, ..., x_{SL}\}$. L'ensemble S doit avoir la capacité d'être un système générateur des données de F. Ainsi, tout vecteur $\phi(x_i) = \phi_i$ de F peut s'écrire sous la forme d'une combinaison linéaire des vecteurs sélectionnés, l'estimé de ϕ_i peut alors s'écrire :

$$\hat{\phi}_i = \phi_S . a_i \tag{1.21}$$

où $\phi_S = (\phi_{S1}, \phi_{S2}, ..., \phi_{SL})$ est la matrice des vecteurs sélectionnés dans F et $a_i = (a_i^1, a_i^2, ..., a_i^L)^t$ les coefficients.

On parle d'estimé parce que pour avoir l'égalité, il faut que le nombre de vecteurs formant S soit supérieur ou égal à la dimension de l'ensemble des données dans F; or, la dimension de F est infinie dans certains cas.

L'estimé $\hat{\phi}_i$ se rapproche de ϕ_i si le ratio ci-après tend vers zéro.

$$\delta_i = \frac{\left\| \phi_i - \hat{\phi}_i \right\|^2}{\left\| \phi_i \right\|^2} \tag{1.22}$$

Pour déterminer les coefficients a_i qui permettent de minimiser δ_i , on pose les dérivées partielles de δ_i par rapport à a_i égales à 0, ce qui conduit à la relation suivante :

$$\min \delta_i = 1 - \frac{K_{Si}{}^t K_{SS}{}^{-1} K_{Si}}{k_{ii}} \tag{1.23}$$

avec $K_{SS} = \begin{pmatrix} k_{S1S1} & k_{S2S1} & \cdots & k_{SLS1} \\ k_{S1S2} & k_{S2S2} & \cdots & k_{SLS2} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ k_{S1SL} & k_{S2SL} & \cdots & k_{SLSL} \end{pmatrix}$ et $K_{Si} = \begin{bmatrix} k_{S1i} \\ k_{S2i} \\ . \\ . \\ . \\ k_{SLi} \end{bmatrix}$

Le but consiste donc à déterminer l'ensemble S des vecteurs sélectionnés qui minimisent (1.23) pour toutes les observations. Ce qui conduit à :

$$\min_S \left(\sum_{x_i \epsilon X} \left(1 - \frac{K_{Si}{}^t K_{SS}{}^{-1} K_{Si}}{k_{ii}} \right) \right) \tag{1.24}$$

La condition exprimée par l'équation 1.24 est réalisée lorsque $\left(\sum_{x_i \epsilon X} \left(\frac{K_{Si}{}^t K_{SS}{}^{-1} K_{Si}}{k_{ii}} \right) \right)$ est maximal, d'où la relation 1.24 est équivalente à :

$$\max_S \left(\sum_{x_i \epsilon X} \left(\frac{K_{Si}{}^t K_{SS}{}^{-1} K_{Si}}{k_{ii}} \right) \right) \tag{1.25}$$

L'algorithme de sélection des vecteurs de S proposé dans [7, 18] est un processus itératif qui a pour but de former l'ensemble S avec critère de maximisation de la fonction objective $J_S = \left(\sum_{x_i \in X} \left(\frac{K_{Si}{}^t K_{SS}{}^{-1} K_{Si}}{k_{ii}} \right) \right)$. Il s'agit d'un problème d'optimisation séquentielle.

Les vecteurs sélectionnés définissent le sous-espace $\mathcal{S}$ de F qui représente le mieux la structure de toutes les données d'apprentissage. On procède à la projection de toutes les observations x_i dans le sous-espace $\mathcal{S}$ par une transformation linéaire :

$$z_i = \phi_S{}^t \phi_i = \left(k(x_{S1}, x_i), ..., k(x_{SL}, x_i) \right) \tag{1.26}$$

Et pour les tâches de prédiction, Baudat et al.[7, 18] proposent d'utiliser une fonction de régression linéaire :

$$\hat{y}_i = z_i{}^t . A + \beta^t \tag{1.27}$$

où les paramètres A et β sont déterminés en minimisant la somme des carrés des écarts résiduels.

1.3.5 Relevance Vector Machine (RVM)

Le RVM est un modèle constitué de somme de fonctions noyau comme le SVM, mais basé sur un traitement bayésien [19, 20]. Soit un problème bi-classe où les observations sont $\{(x_1, t_1), ..., (x_\ell, t_\ell)\}$ avec $x_i \in R^d$ et $t_i \in \{0,1\}$. Le RVM utilise une combinaison linéaire de fonction noyaux définie comme suit :

$$y(x, w) = \sum_{i=1}^{\ell} w_i k(x, x_i) + w_0 \tag{1.28}$$

En utilisant la fonction logistique $\sigma(y)=1/(1+e^{-y})$ avec une distribution de Bernoulli pour la probabilité conditionnelle $P(t/x)$, la vraisemblance suivante peut s'écrire :

$$P(t/w)=\prod_{i=1}^{\ell}\sigma\{y(x_i,w)\}^{t_i}\left[1-\sigma\{y(x_i,w)\}\right]^{1-t_i} \tag{1.29}$$

Les poids w coefficients des fonctions noyaux (équation 1.28) sont estimés par une loi gaussienne de moyenne nulle mais avec différentes valeurs de déviation, comme le montre l'équation 1.30 :

$$P(w/\alpha)=\prod_{j=0}^{\ell}N(w_i/0,\alpha_i^{-1}) \tag{1.30}$$

avec $\alpha=(\alpha_0,\alpha_1,...,\alpha_\ell)$ et $w=(w_0,w_1,...,w_\ell)$

L'utilisation de l'approximation basée sur la méthode de Laplace permet de déterminer les valeurs des poids w_i qui maximisent la probabilité $P(w/t,\alpha)$. La procédure itérative d'approximation de Laplace est décrite ci-après en deux points :

1. En fonction des valeurs courantes de α, déterminer les valeurs des poids w_{MP} les plus probables par la maximisation de $P(w/t,\alpha)$.

Puisque $p(w/t,\alpha)\propto P(t/w)p(w/\alpha)$, alors il suffit de trouver les poids w qui maximisent :

$$\log\{P(t/w)p(w/\alpha)\}=\sum_{i=1}^{\ell}[t_i\log y_i+(1-t_i)\log(1-y_i)]-\frac{1}{2}wAw \tag{1.31}$$

avec $y_i=\sigma\{y(x_i;w)\}$ et $A=diag(\alpha_0,\alpha_1,...,\alpha_l)$.

En considérant l'ordre 2 de la dérivée de la fonction objective (1.31), nous avons :

$$\nabla_w \nabla_w \log p(w/t,\alpha) \mid w_{MP} = -(\Phi^T B \Phi + A) \tag{1.32}$$

où $B = diag(\beta_0, \beta_1, ..., \beta_\ell)$ avec $\beta_i = \sigma\{y(x_i)\}\left[1-\sigma\{y(x_i)\}\right]$

et $\Phi = \left[\phi(x_1), \phi(x_2), ..., \phi(x_\ell)\right]^T$

De la relation 1.32 et du fait que $\nabla_w \log p(w/t,\alpha) \mid w_{MP} = 0$, nous pouvons écrire :

$$\sum = (\Phi^T B \Phi + A)^{-1} \tag{1.33}$$

et

$$w_{MP} = \sum \Phi^T B t \tag{1.34}$$

2. En utilisant les valeurs de $\sum$ et w_{MP}, les hyper-paramètres α_i sont mis à jour en utilisant la formule suivante :

$$\alpha_i^{new} = \frac{\gamma_i}{w_{MP}^i} \tag{1.35}$$

avec $\gamma_i = 1 - \alpha_i \sum_{ii}$

1.4 Principe et Modélisation des SVM

1.4.1 Risque structurel et dimension VC

Le but de l'apprentissage est d'estimer l'application $f : x_i \mapsto y_i$ la mieux adaptée à la classification ou à la régression en tenant compte des observations d'apprentissage. Souvent, on cherche la fonction qui permet de minimiser uniquement l'erreur d'apprentissage appelée le risque empirique.

$$R_{emp}[f] = \frac{1}{\ell} \sum_{i=1}^{\ell} \frac{1}{2} \left| f(x_i) - y_i \right| \tag{1.36}$$

Mais la minimisation de 1.36 ne garantit pas toujours le risque minimum qui désigne l'erreur en test [21]. Considérons l'exemple illustré en figure 5 où nous avons représenté trois fonctions de décision pour un problème bi-classe.

Le modèle linéaire est en situation de sous-apprentissage tandis que le modèle de haut degré est en situation de sur-apprentissage, car l'objet x lors de la classification sera mal classé. Un compromis entre ces deux modèles est la fonction de degré intermédiaire représentée.

Pour éviter de choisir les fonctions de décision issues de sur-apprentissage, Vapnik [21, 22] introduit la notion du risque structurel dont la minimisation conjointe avec le risque empirique permet de sélectionner dans l'espace des fonctions H, une fonction de décision qui minimise l'erreur en généralisation. Le risque structurel est donc une mesure de la complexité de la fonction affectant sa capacité de généralisation en test.

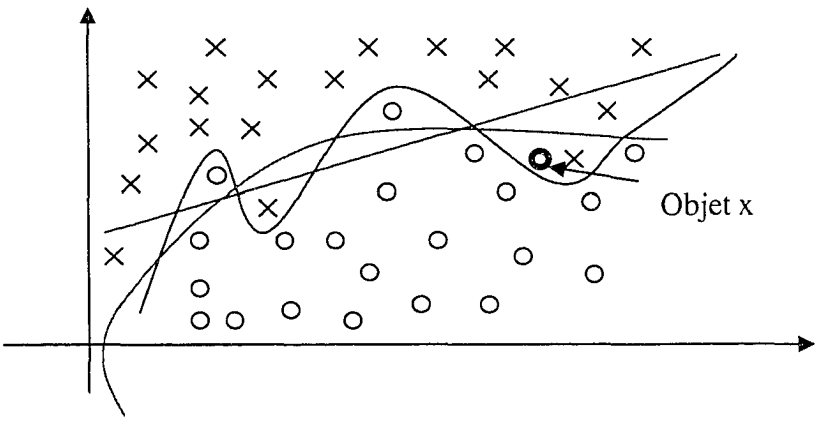


Figure 5 Phénomènes de sous-apprentissage et de sur-apprentissage

Ainsi, l'estimation de la fonction f ayant un fort pouvoir de généralisation dépend du risque empirique et du risque structurel. Mais, contrairement au risque empirique, le

risque structurel est difficile à quantifier, d'où la nécessité de l'utilisation de la notion de dimension VC (Vapnik-Chervonenkis).

Dans [23, 24], Vapnik et Chervonenkis définissent la dimension VC pour un ensemble de fonctions comme suit : la dimension VC d'un ensemble de fonctions $Q(z,\alpha)$, $\alpha \in \Lambda$ où Λ désigne l'ensemble des paramètres, est le nombre maximal h de vecteurs $z_1,...,z_h$ qui peuvent être séparés en deux classes selon toutes les 2^h possibilités en utilisant l'ensemble de fonctions.

Par exemple, considérons l'espace des données $X \in R^2$ et l'ensemble des fonctions linéaires de R^2, une droite ne peut diviser au plus 3 points selon toutes les possibilités de bi-classe. Lorsqu'on a 4 points, une droite est incapable de faire la séparation selon toutes les possibilités de classification binaire (voir figure 6).

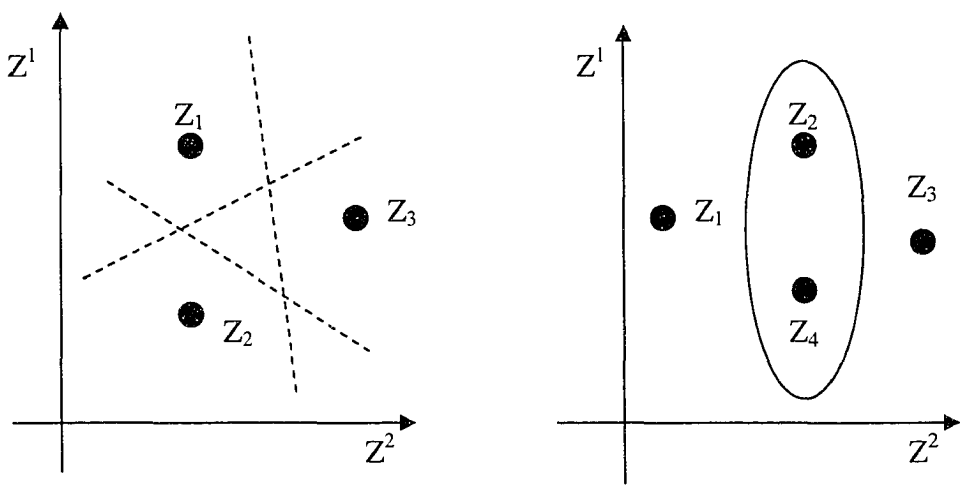


Figure 6 : Dimension VC de l'ensemble des fonctions linéaire de R^2 : la dimension VC est égale à 3. Pour 4 points, il n'est pas possible de séparer les éléments z_2, z_4 par une droite des éléments z_1, z_3 .(Extraite de [25])

Connaissant la dimension VC, le vrai risque (l'erreur en généralisation) R peut être mesuré sur l'ensemble d'apprentissage [25]. Ainsi, on a avec la probabilité $1-\eta$:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{1}{\ell}\left(h \log \frac{2\ell}{h} + 1 - \log \frac{\eta}{4}\right)} \tag{1.37}$$

On notera que si la taille de l'ensemble d'apprentissage est très large et la dimension VC est finie, le terme de pénalisation exprimant le risque structurel devient négligeable. Ainsi, nous avons une très bonne généralisation en minimisant uniquement le risque empirique.

1.4.2 Principe des SVM : maximisation de la marge de séparation

Supposons que les données d'apprentissage $\{(x_1, y_1), ..., (x_\ell, y_\ell)\}$ avec $x_i \in R^d$ et $y_i \in \{-1, 1\}$ sont linéairement séparables par l'hyperplan :

$$w.x + b = 0 \tag{1.38}$$

L'hyperplan $(w^*.x) + b = 0$ avec $|w^*| = 1$ qui permet de classifier tous les vecteurs d'apprentissage comme suit :

$$y = \begin{cases} +1 & si \quad (w^*.x) + b \geq \Delta \\ -1 & si \quad (w^*.x) + b \leq \Delta \end{cases} \tag{1.39}$$

est appelé «Δ-margin separating hyperplane».

Cet hyperplan particulier permet d'énoncer le théorème [25] ci-après :
Si l'ensemble des observations d'apprentissage X appartient au cercle de rayon R, alors h la dimension VC de l'ensemble des fonctions de «Δ-margin separating hyperplane» est bornée par :

$$h \leq \min\left(\left[\frac{R^2}{\Delta^2}\right], d\right) + 1 \tag{1.40}$$

Ainsi lorsque Δ est bien choisi, on arrive à avoir $h \leq d+1$.

On définit alors l'hyperplan optimal comme le « Δ-margin separating hyperplane» avec $\Delta = 1/|w^*|$. Ainsi pour minimiser la dimension VC, il faut que sa borne supérieure soit minimale. Avec d fixe, dimension de l'espace des observations, il faut choisir l'hyperplan qui maximise la marge Δ et nous avons donc une séparation sans erreur avec la distance de l'observation la plus proche de l'hyperplan maximale (voir figure 16). L'équation 1.39 devient alors :

$$y_i\left[(w^*.x_i)+b\right] \geq 1, \quad i = 1,...,l \tag{1.41}$$

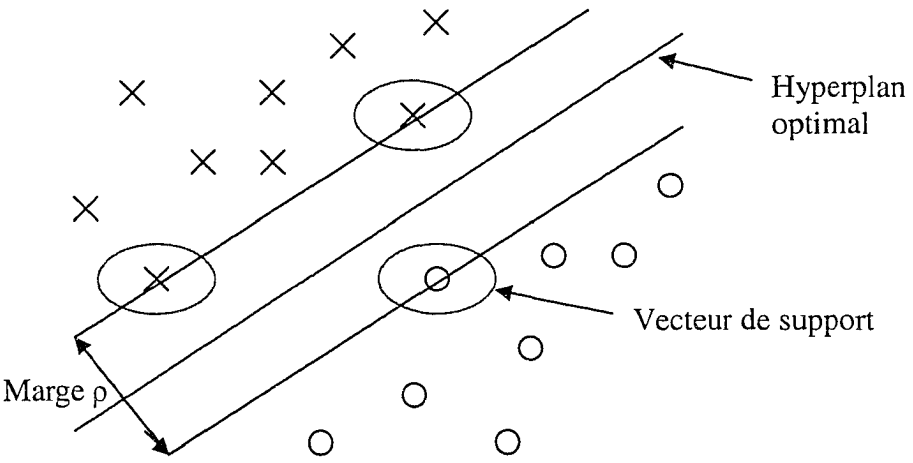


Figure 7 Représentation de l'hyperplan et des vecteurs de support

Maximiser la marge ρ (la plus petite distance entre les deux classes et l'hyperplan) est équivalent à maximiser la somme des distances par rapport à l'hyperplan.

$$\rho = \min_{x_i / y_i = 1} \frac{w.x_i + b}{\|w\|} - \max_{x_i / y_i = -1} \frac{w.x_i + b}{\|w\|} = \frac{2}{\|w\|} \tag{1.42}$$

Par conséquent, l'hyperplan optimal défini par (ω_0,b_0) est celui qui satisfait la condition (1.41) et qui minimise $G(w)=\frac{\|w\|^2}{2}$ lorsque les données sont linéairement séparables.

1.4.3 SVM linéaire et séparable

Pour construire l'hyperplan séparable dans le cas où les données sont linéairement séparables, on résout le problème d'optimisation ci-après :

$$\text{Minimiser } G(w,b)=\frac{1}{2}(w.w) \tag{1.43}$$

avec les l contraintes d'inégalités : $y_i\left[(x_i.w)+b\right]\geq 1, \qquad i=1,...,l$

Le lagrangien du problème 1.43 est donné par :

$$L(w,b,\alpha)=\frac{1}{2}w.w-\sum_{i=1}^{\ell}\alpha_i\left\{y_i\left[(x_i.w)+b\right]-1\right\} \tag{1.44}$$

où α_i sont les multiplicateurs de Lagrange, $\alpha_i\geq 0$.

La résolution de ce problème d'optimisation, après avoir introduit les multiplicateurs de Lagrange, est équivalente à déterminer les valeurs de w, de b et de α_i qui vérifient :

$$\frac{\partial L(w,b,\alpha)}{\partial w}=0 \tag{1.45}$$

$$\frac{\partial L(w,b,\alpha)}{\partial b}=0 \tag{1.46}$$

$$\alpha_i\left\{y_i\left[(x_i.w)+b\right]-1\right\}=0 \quad avec \quad i=1,...,\ell \tag{1.47}$$

$$y_i\left[(x_i.w)+b\right]-1\geq 0 \quad et \quad \alpha_i\geq 0 \quad avec \quad i=1,...,\ell \tag{1.48}$$

Les équations 1.45 et 1.46 permettent de déduire respectivement que :

$$w = \sum_{i=1}^{\ell} \alpha_i y_i x_i \tag{1.49}$$

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0 \tag{1.50}$$

En considérant l'équation 1.47, nous pouvons dire que lorsque les données ne sont pas sur la marge, c'est-à-dire $y_i[(x_i.w)+b]-1>0$, les multiplicateurs de Lagrange correspondants sont nuls. Ainsi, seules les données se trouvant sur la marge, c'est-à-dire celles qui saturent la condition 1.48, ont des multiplicateurs non nuls et par conséquent définissent la valeur de *w* selon 1.49. On les appelle *vecteurs de support.*

En remplaçant les résultats 1.49 et 1.50 dans l'équation 1.44, on obtient un problème d'optimisation quadratique :

$$\text{Maximiser} \quad W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j (x_i.x_j) \tag{1.51}$$

sous les contraintes $\sum_{i=1}^{\ell} \alpha_i y_i = 0 \quad et \quad \alpha_i \geq 0, \quad i = 1,...,\ell$

La résolution de ce problème quadratique dont les inconnues sont les α_i permet de déterminer l'équation de l'hyperplan optimal :

$$w_0.x + b_0 = 0 \tag{1.52}$$

avec

$$w_0 = \sum_{i=1}^{\ell} \alpha_i^0 y_i x_i \tag{1.53}$$

où α_i^0 sont les solutions de 1.51

et

$$b_0 = \frac{1}{2}\left[\left(w_0.x^*(1)\right)+\left(w_0.x^*(-1)\right)\right] \tag{1.54}$$

où $x^*(1)$ est un vecteur de support de la classe positive et $x^*(-1)$ de la classe négative.

1.4.4 SVM linéaire et marge molle (cas non séparable)

Dans la section précédente, nous avons supposé que les données sont parfaitement linéairement séparables. Mais si cela n'est pas le cas, il faut relaxer les inégalités décrites par 1.41 en introduisant des variables d'écart $\xi_i \geq 0$. Les contraintes d'optimisation deviennent alors :

$$y_i\left[(x_i.w)+b\right] \geq 1-\xi_i, \qquad i=1,...,\ell \tag{1.55}$$

Les variables d'écart permettent de quantifier la mauvaise position des données. Il faut alors aussi minimiser $\sum \xi_i$ ou $\sum \xi_i^2$ dans la fonction objective déterminant l'hyperplan, afin de minimiser l'erreur globale. Ainsi, nous avons deux types de problèmes d'optimisation pour le cas non séparable :

Norme 1 :

$$\text{Minimiser } G(w,b,\xi) = \frac{1}{2}(w.w) + C\sum_{i=1}^{\ell}\xi_i \tag{1.56}$$

avec les $2l$ contraintes d'inégalités : $y_i\left[(x_i.w)+b\right] \geq 1-\xi_i, \quad \xi_i \geq 0, \qquad i=1,...,\ell$

où C est une valeur positive donnée pour pénaliser les variables d'écarts.

Norme 2 :

$$\text{Minimiser } G(w,b,\xi)=\frac{1}{2}(w.w)+C\sum_{i=1}^{\ell}\xi_i^2 \tag{1.57}$$

avec les $2l$ contraintes d'inégalités : $y_i\left[(x_i.w)+b\right]\geq 1-\xi_i,\quad \xi_i\geq 0,\qquad i=1,...,\ell$

où C est une valeur positive donnée pour pénaliser les variables d'écarts.

Dans la plupart du temps, la norme 1 est utilisée pour définir le problème des SVM pour la marge molle, c'est-à-dire avec des données qui peuvent se trouver dans la marge ou être mal classées. Dans ce qui suit, nous décrivons en bref la résolution du problème d'optimisation avec la norme 1.

Le Lagrangien du problème 1.56 s'écrit :

$$L(w,b,\xi,\alpha,\lambda)=\frac{1}{2}w.w+C\sum_{i=1}^{\ell}\xi_i-\sum_{i=1}^{\ell}\alpha_i\left\{y_i\left[(x_i.w)+b\right]-1+\xi_i\right\}-\sum_{i=1}^{\ell}\lambda_i\xi_i \tag{1.58}$$

où $\alpha_i\geq 0$ et $\lambda_i\geq 0$ sont les multiplicateurs de Lagrange.

En appliquant les théorèmes de différentiation, nous avons :

$$\frac{\partial L(w,b,\xi,\alpha,\lambda)}{\partial w}=0 \tag{1.59}$$

$$\frac{\partial L(w,b,\xi,\alpha,\lambda)}{\partial b}=0 \tag{1.60}$$

$$\frac{\partial L(w,b,\xi,\alpha,\lambda)}{\partial \xi_i}=0 \tag{1.61}$$

$$\alpha_i\left\{y_i\left[(x_i.w)+b\right]-1+\xi_i\right\}=0 \quad avec \quad i=1,...,\ell \tag{1.62}$$

$$\lambda_i\xi_i=0 \quad avec \quad i=1,...,\ell \tag{1.63}$$

$$y_i\left[(x_i.w)+b\right]-1+\xi_i\geq 0 \quad et \quad \alpha_i\geq 0 \quad avec \quad i=1,...,\ell \tag{1.64}$$

$$\xi_i\geq 0 \quad et \quad \lambda_i\geq 0 \quad avec \quad i=1,...,\ell \tag{1.65}$$

À partir des équations 1.59 et 1.60, on retrouve les égalités 1.49 et 1.50. De plus les équations découlant de 1.61 permettent d'écrire :

$$C - \alpha_i - \lambda_i = 0 \quad avec \quad i = 1,...,\ell \tag{1.66}$$

Ainsi, on a : $\lambda_i = C - \alpha_i, \quad \forall i = 1,...,\ell$

Or $\lambda_i \geq 0$

Donc $C - \alpha_i \geq 0, \quad \forall i = 1,...,\ell$

D'où :

$$0 \leq \alpha_i \leq C, \quad \forall i = 1,...,\ell \tag{1.67}$$

L'équation 1.65 combinée avec 1.66 permet d'écrire :

$$\xi_i (\alpha_i - C) = 0 \quad pour \quad i = 1,...,\ell \tag{1.68}$$

Et nous pouvons alors en déduire de 1.68 que pour toutes les données de variables d'écart non nulles, $\alpha_i = C$. Ainsi, les données qui sont mal classées au-delà de la marge ou sont dans la marge ont des multiplicateurs $\alpha_i = C$.

De même, avec l'équation 1.62, nous avons la même conclusion que celle obtenue dans la section précédente, c'est-à-dire que les données qui sont bien classées en dehors de la marge, $y_i \left[(x_i . w) + b \right] - 1 + \xi_i > 0$, ont des multiplicateurs $\alpha_i = 0$.

En somme, les conditions KKT permettent de regrouper les données d'apprentissage en trois ensembles :

- l'ensemble des données bien classées hors de la marge, $\alpha_i = 0$.
- l'ensemble des données mal classées selon la marge, $\alpha_i = C$.
- l'ensemble des données situées sur la marge, $0 < \alpha_i < C$.

En remplaçant les divers résultats obtenus précédemment dans l'équation 1.58, on obtient le même problème d'optimisation quadratique que dans le cas séparable, mais avec des contraintes différentes :

$$\text{Maximiser} \quad W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j (x_i . x_j) \tag{1.69}$$

$$\text{Sous les contraintes} \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0 \quad et \quad 0 \le \alpha_i \le C, \quad i = 1, ..., \ell$$

Et la résolution de ce problème fournit les coefficients α_i^0 pour définir le paramètre $w_0 = \sum_{i=1}^{\ell} \alpha_i^0 y_i x_i$ l'équation de l'hyperplan.

1.4.5 SVM non linéaire : "astuce du noyau" («kernel trick»)

Généralement, en reconnaissance de formes, nous n'avons pas souvent des problèmes linéaires. Ainsi, pour étendre l'utilisation des machines à vecteurs de support, on a introduit la technique de l'espace augmenté. Cette technique repose sur l'application de la méthode des noyaux appelée "astuce du noyau".

Puisque dans les formules de construction des SVM, seul intervient le produit scalaire de deux points, on peut donc utiliser toute fonction noyau $k(x, y)$ respectant les conditions de Mercer afin de garantir la semi-positivité de la matrice de la fonction objective. Ainsi le problème quadratique à résoudre devient :

$$\text{Maximiser} \quad W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j k(x_i . x_j) \tag{1.70}$$

$$\text{Sous les contraintes} \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0 \quad et \quad 0 \le \alpha_i \le C, \quad i = 1, ..., \ell$$

Et pour toute observation z de test, nous avons la fonction de décision qui s'écrit :

$$g(x) = sign\left(\sum_{i=1}^{\ell} y_i \alpha_i k(x_i, z) + b\right) \tag{1.71}$$

1.5 Algorithmes d'apprentissage des SVM

1.5.1 Méthode de décomposition

L'apprentissage d'une machine à vecteurs de support revient à résoudre un problème d'optimisation quadratique. Mais lorsque la taille de l'ensemble d'apprentissage devient très importante, les algorithmes classiques de résolution de problème QP sont inefficaces. Nous avons alors besoin de techniques spécifiques. Ainsi, la méthode de décomposition proposée par Osuna et al.[26] permet de faire l'apprentissage des SVMs lorsque la taille de l'ensemble d'apprentissage est grande.

De façon matricielle, le problème QP se pose comme suit :

$$\max_{\Lambda} \quad W(\Lambda) = \Lambda^T 1 - \frac{1}{2}\Lambda^T D \Lambda \tag{1.72}$$

sous les contraintes :

$$\Lambda^T Y = 0$$
$$\Lambda - C1 \leq 0$$
$$-\Lambda \leq 0$$

où $(1)_i = 1$, $D_{ij} = y_i y_j k(x_i, x_j)$, $\Lambda^T = (\alpha_1, ..., \alpha_\ell)$ et $Y^T = (y_1, ..., y_\ell)$.

Les conditions de KKT d'optimalité démontrées dans la section précédente permettent d'écrire les conditions nécessaires et suffisantes:

$$\begin{aligned} 0<\alpha_i<C \quad & \text{si} \quad y_i f(x_i)=1 \\ \alpha_i=C \quad & \text{si} \quad y_i f(x_i)\le 1 \\ \alpha_i=0 \quad & \text{si} \quad y_i f(x_i)\ge 1 \end{aligned} \tag{1.73}$$

avec

$$f(x_i)=\sum_{j=1}^{\ell} y_j \alpha_j k(x_i,x_j)+b \tag{1.74}$$

Donc, si les valeurs des composantes de Λ satisfont aux conditions 1.73, nous avons résolu le problème 1.72.

La stratégie de décomposition consiste à partitionner l'ensemble d'apprentissage en deux sous-ensembles *B* et *N*, où *B* est appelé l'ensemble actif. Ainsi, on décompose aussi le vecteur Λ en deux vecteurs Λ_B et Λ_N. On forme alors un sous problème QP dont la variable est le vecteur Λ_B et on garde fixe le vecteur Λ_N.

$$\begin{aligned} \max_{\Lambda_B} \quad & W(\Lambda_B) = \Lambda_B^T 1+\Lambda_N^T 1-\frac{1}{2}\left[\Lambda_B^T D_{BB}\Lambda_B+\Lambda_B^T D_{BN}\Lambda_N+\Lambda_N^T D_{NB}\Lambda_B+\Lambda_N^T D_{NN}\Lambda_N\right] \\ & \Lambda_B^T Y_B+\Lambda_N^T Y_N=0 \\ \text{sous les contraintes :} \quad & \Lambda_B - C1\le 0 \\ & -\Lambda_B\le 0 \end{aligned} \tag{1.75}$$

Puisque les termes $\Lambda_N^T 1$ et $\Lambda_N^T D_{NN}\Lambda_N$ sont constants par rapport au sous-problème et que la fonction est symétrique, le problème 1.75 peut encore s'écrire :

$$\max_{\Lambda_B} \quad W(\Lambda_B) = \Lambda_B^T(1-D_{BN}\Lambda_N)-\frac{1}{2}\Lambda_B^T D_{BB}\Lambda_B \tag{1.76}$$

sous les contraintes : $\Lambda_B^T Y_B+\Lambda_N^T Y_N=0, \quad \Lambda_B - C1\le 0 \quad et \quad -\Lambda_B\le 0$

La technique pour optimiser le problème 1.72 est d'optimiser itérativement les sous-problèmes 1.76 et à chaque itération de remplacer un élément de l'ensemble B par un élément de N qui viole l'une des conditions de KKT (1.73). La figure suivante montre l'algorithme présenté dans [26].

1. Choisir arbitrairement les éléments de la base d'apprentissage pour former l'ensemble B et le reste pour former l'ensemble N.
2. Résoudre le sous problème défini par les variables de Λ_B
3. Tant qu'il existe $j \in N$ tel que :
   * $\alpha_j = 0$ et $y_j f(x_j) < 1$
   * $\alpha_j = C$ et $y_j f(x_j) > 1$
   * $0 < \alpha_j < C$ et $y_j f(x_j) \neq 1$

   Remplacer n'importe quel élément $i \in B$ par l'élément j trouvé et résoudre le nouveau sous-problème.

Figure 8 Algorithme de décomposition pour l'apprentissage des SVMs (extraite de [26])

1.5.2 Algorithme de Joachims : méthode de décomposition améliorée

La méthode de décomposition proposée par Osuna et al.[26] permet de résoudre le problème de large base de données. L'algorithme converge après un nombre fini d'itérations, mais il requiert un temps CPU important. Et c'est pour réduire ce temps de calcul que Joachims[27] a développé SVM^{light} qui est une technique basée sur la décomposition.

L'algorithme de Joachims diffère de celui d'Osuna et al. par la sélection des q éléments de l'ensemble actif B. Pour accélérer la convergence, Joachims propose d'utiliser une stratégie basée sur la méthode de Zoutendijk[28]. L'idée est de trouver la direction admissible d pour la descente de gradient et de former à partir de cette direction l'ensemble B. La détermination de d revient à résoudre un autre problème d'optimisation avec des contraintes :

$$\min_{d} \quad V(d) \quad = g(\alpha^{(t)})^T d$$

$$\text{sous les contraintes :} \quad \begin{cases} y^T d = 0 \\ d_i \geq 0 \quad pour \quad i : \alpha_i = 0 \\ d_i \leq 0 \quad pour \quad i : \alpha_i = C \\ -1 \leq d \leq 1 \\ \left| \{ d_i : d_i \neq 0 \} \right| = q \end{cases} \tag{1.77}$$

La résolution de ce précédent problème d'optimisation permet de trouver le vecteur d, avec q composantes non nulles, qui permet de garantir la convergence rapide du problème 1.72 à partir de la résolution du sous-problème 1.76. Les indices des q composantes non nulles sont ceux utilisés pour former l'ensemble actif B.

1.5.3 Optimisation séquentielle minimale : SMO

Le SMO développé par Platt[29] est une variante de la technique de décomposition où l'ensemble actif est formé de deux éléments. Ainsi, à chaque itération on résout un problème d'optimisation à deux variables qui se fait de façon algébrique : deux multiplicateurs α_i sont sélectionnés à partir des heuristiques et on procède à leurs mises à jour en fixant les autres valeurs de α.

Désignons par α_1 et α_2 les deux multiplicateurs de Lagrange à mettre à jour. En considérant les contraintes linéaires du problème, on détermine des bornes suivantes pour l'élément α_2 à déterminer :

$$L \leq \alpha_2 \leq H \tag{1.78}$$

Avec

$L = \max(0, \alpha_2^{old} - \alpha_1^{old})$ et $H = \min(C, C + \alpha_2^{old} - \alpha_1^{old})$ si $y_1 y_2 = -1$

$L = \max(0, \alpha_2^{old} + \alpha_1^{old} - C)$ et $H = \min(C, \alpha_2^{old} + \alpha_1^{old})$ si $y_1 y_2 = 1$

Les bornes L et H ainsi déterminées permettent de réévaluer la valeur de α_2 après avoir appliqué la formule suivante, découlant de la maximisation de la fonction $W(\alpha_1, \alpha_2)$:

$$\alpha_2^{new} = \alpha_2^{old} - \frac{y_2(E_1 - E_2)}{\eta} \tag{1.79}$$

où $E_i = f^{old}(x_i) - y_i$ et $\eta = 2k(x_1, x_2) - k(x_1, x_1) - k(x_2, x_2)$

Et la réévaluation de la valeur de α_2 se fait en utilisant l'équation 1.80

$$\alpha_2^{new*} = \begin{cases} H & si \quad \alpha_2^{new} \geq H \\ \alpha_2^{new} & si \quad L < \alpha_2^{new} < H \\ L & si \quad \alpha_2^{new} \leq L \end{cases} \tag{1.80}$$

Après avoir déterminé α_2, le calcul de α_1 est effectué en utilisant la contrainte linéaire $y_1\alpha_1^{new} + y_2\alpha_2^{new} = y_1\alpha_1^{old} + y_2\alpha_2^{old}$ et on en déduit alors que :

$$\alpha_1^{new} = \alpha_1^{old} + s(\alpha_2^{old} - \alpha_2^{new*}) \tag{1.81}$$

avec $s = y_1 y_2$

Le SMO, comme proposé par Platt dans [29], est une procédure très lente. Et ceci dépend surtout des heuristiques utilisées pour sélectionner les deux multiplicateurs à mettre à jour. Dans [30], Keerthi et al ont proposé une technique de sélection des deux multiplicateurs qui permet une rapide convergence de l'algorithme.

1.6 Classification multi-classe avec les SVM

Les machines à vecteurs de support traitent habituellement des problèmes bi-classes. Cependant, il existe des techniques de combinaison pour résoudre des problèmes multi-classes. Deux types d'approches sont souvent utilisés pour réaliser des classifieurs multi-classes à base des SVMs :

- l'approche un-contre-un
- l'approche un-contre-tous

1.6.1 Approche un-contre-tous

L'idée est de construire autant de SVMs que de classes où chaque SVM permet de séparer une classe de toutes les autres. Ainsi, pour un problème à c classes, il faut entraîner c SVMs qui seront ensuite couplés pour prendre des décisions lors du test. Le couplage naïf qui consiste à attribuer à une observation la classe dont la sortie du SVM est positive, n'est pas satisfaisant. Car pour certaines observations ambiguës, plusieurs sorties peuvent être positives.

Pour réaliser un bon couplage, il est recommandé de normaliser la sortie des SVMs ou de les convertir en mesure de probabilités. Ainsi, un exemple est associé à la classe du SVM ayant la plus forte valeur de sortie normalisée ou de probabilité.

1.6.2 Approche un-contre-un

Cette démarche requiert la construction de $c(c-1)/2$ SVMs pour un problème de c classes. Chaque SVM permet de traiter un problème bi-classe formé des données d'un couple (ω_i, ω_j) de classes. En test, chaque sortie des SVM convertie en probabilité, fournit pour un exemple x donné :

$$p_{ij} = P(x \in \omega_i \mid x, x \in \omega_i \cup \omega_j) \tag{1.82}$$

La règle de décision est alors donnée par :

$$\arg \max_{1 \le i \le c} \hat{p}_i \tag{1.83}$$

avec

$$\hat{p}_i = \frac{2}{c(c-1)} \sum_{i \neq j} \sigma(\hat{p}_{ij}) \tag{1.84}$$

où σ désigne la fonction de couplage.

Plusieurs types de couplage sont rapportés dans la littérature {Hastie, 1998 #17;Wu, 2004 #16;Moreira, 1998 #1;Hsu, 2002 #2;Wu, 2004 #16}, mais il demeure toujours un vif débat sur le modèle de couplage le plus efficace. Dans [33], les fonctions de couplage du tableau I sont rapportées, avec différents commentaires.

Cette dernière approche est le plus souvent utilisée à cause des problèmes de ressources que requiert l'approche "un-contre-tous". Car pour l'apprentissage des SVM dans l'approche "un-contre-tous", il faut utiliser toute la base de données de toutes les classes.

Tableau I

Fonctions de couplage

Fonctions	Expressions
PW1	$\sigma(x)=\begin{cases}1 & si \quad x\geq 0.5\\ 0 & \text{sinon}\end{cases}$
PW2	$\sigma(x)=x$
PW3	$\sigma(x)=\dfrac{1}{1+\exp[-12(x-0.5)]}$
PW4	$\sigma(x)=\begin{cases}1 & si \quad x\geq 0.5\\ x & \text{sinon}\end{cases}$
PW5	$\sigma(x)=\begin{cases}x & si \quad x\geq 0.5\\ 0 & \text{sinon}\end{cases}$

1.7 Conclusion

Dans ce chapitre, nous avons présenté la méthode des noyaux et la description sommaire de certains algorithmes de classification utilisant cette méthode. Enfin, nous avons exposé les machines à vecteurs de support, depuis leur genèse basée sur la minimisation du risque structurel quantifié par la dimension VC.

Le chapitre suivant, nous présentera les diverses techniques développées pour améliorer la performance des SVM, en faisant une bonne sélection de modèle.

CHAPITRE 2

SÉLECTION DE MODÈLE POUR LES SVM: ÉTAT DE L'ART

2.1 Introduction

Plusieurs recherches se sont intéressées très tôt à la problématique de la sélection de modèle pour les machines à vecteurs de support. Et en dehors de la procédure classique de la validation croisée, d'autres critères d'optimisation des hyper-paramètres des SVM ont été développés en se basant sur le «leave-one-out». La procédure de "leave-one-out" est un cas particulier de la validation croisée et elle permet de donner une estimation non biaisée de l'erreur de généralisation.

2.2 Technique de la validation croisée

2.2.1 Théorie de la validation croisée

La validation croisée est une bonne technique d'évaluation de la performance en généralisation d'un classifieur. L'idée derrière cette technique est de tester le modèle du classifieur obtenu sur des données qui n'ont pas participé à l'apprentissage, afin de permettre de prédire le comportement du classifieur face aux nouvelles données.

En pratique, on parle de «*K-fold cross validation*» qui consiste à diviser l'ensemble de validation en k sous-ensembles. On utilise alors un sous-ensemble pour tester le modèle issu de l'apprentissage effectué avec les k-1 autres sous-ensembles. Et on répète k fois cette opération à travers tous les k sous ensembles formés. La moyenne des erreurs déterminées lors des k opérations permet de donner une estimation de la capacité de généralisation du classifieur. La variance des résultats obtenus est réduite lorsque k croît, mais l'inconvénient, c'est le temps de calcul qui devient très long.

Le cas extrême du choix de k pour minimiser la variance a donné naissance à la technique appelée "leave-one-out" (LOO). Dans ce cas, l'entier k vaut la taille totale de l'ensemble d'apprentissage disponible et chaque sous ensemble est constitué alors d'un seul élément. Ainsi, pour chaque opération, on retire une observation de l'ensemble sur laquelle le modèle appris sera testé.

En appliquant le LOO à une machine à vecteurs de support, le nombre d'erreurs de classification vaut :

$$\sum_{p=1}^{\ell} \Psi(-y_p f^p(x_p)) = \sum_{p=1}^{\ell} \Psi\left(-y_p f^0(x_p) + y_p (f^0(x_p) - f^p(x_p))\right) \tag{2.1}$$

où f^0 désigne le classifieur obtenu avec la totalité des observations, f^p représente le classifieur obtenu lorsque la p-ième observation est retirée et Ψ est la fonction échelon définie par $\Psi(x) = 1$ si $x > 0$ et $\Psi(x) = 0$ sinon.

2.2.2 Validation croisée généralisée approchée (GACV)

Wahba et al. [35] ont développé une approximation de la validation croisée. Cette méthode provient des travaux de sélection de modèle dans l'espace de Hilbert en utilisant un critère de généralisation basé sur la distance GCKL (Generalized Comparative Kullback Leibler Distance). Les auteurs de GACV définissent la fonction exprimant LOO comme :

$$V_0(\lambda) = \frac{1}{\ell} \sum_{i=1}^{\ell} g(y_i f_{\lambda}^i(x_i)) \tag{2.2}$$

Et en prenant $g(x) = \text{sgn}(1-x)$, ils ont approximé l'erreur LOO par :

$$GACV(\lambda) = \frac{1}{\ell}\sum_{i=1}^{\ell} g(y_i f_\lambda(x_i)) + \hat{D}(\lambda) \tag{2.3}$$

où $\hat{D}(\lambda) = \frac{1}{\ell}\left[2 \sum_{y_i f_\lambda(x_i)<-1} \frac{\alpha_i}{2\ell\lambda} k(x_i, x_i) + \sum_{y_i f_\lambda(x_i)\in[-1,1]} \frac{\alpha_i}{2\ell\lambda} k(x_i, x_i) \right]$

Le GACV est alors défini pour les SVMs par :

$$GACV(\lambda) = \frac{1}{\ell}\left[\sum_{i=1}^{\ell} \xi_i + 2 \sum_{y_i f_\lambda(x_i)<-1} \alpha_i k(x_i, x_i) + \sum_{y_i f_\lambda(x_i)\in[-1,1]} \alpha_i k(x_i, x_i) \right] \tag{2.4}$$

Dans [35], les auteurs ont démontré que la minimisation du GACV permet de réaliser une bonne sélection de modèle pour les SVMs.

2.3 Bornes de "leave-one-out" pour les SVM

2.3.1 Nombre de vecteurs de support

Puisque seuls les points dont les multiplicateurs de Lagrange sont non nuls, c'est-à-dire les vecteurs de support, permettent de définir un classifieur SVM, alors en retirant les points non vecteurs de support de l'ensemble d'apprentissage, la fonction de décision obtenue n'est pas modifiée. Ainsi, nous avons $f^0(x_p) = f^p(x_p)$ pour toutes les observations qui ne sont pas vecteurs de support. Par conséquent, la borne supérieure de l'erreur produite au cours de la procédure LOO ne peut alors pas dépasser le nombre de vecteurs de support [25].

$$T = \frac{N_{VS}}{\ell} \tag{2.5}$$

2.3.2 Borne de Jaakkola-Haussler

Jaakkola et Haussler dans [36] ont démontré que l'inégalité ci-après est vraie lorsque nous avons une machine à vecteur de support sans biais :

$$y_p\left(f^0(x_p)-f^p(x_p)\right)\leq \alpha_p^0 k(x_p,x_p) \tag{2.6}$$

Nous pouvons alors en déduire que la borne supérieure de l'erreur en généralisation selon la procédure LOO vaut :

$$T=\frac{1}{\ell}\sum_{p=1}^{\ell}\Psi(\alpha_p^0 k(x_p,x_p)-1) \tag{2.7}$$

2.3.3 Borne de Opper-Winther

Opper et Winther ont proposé dans [37, 38] une technique pour la classification binaire avec le noyau gaussien en se basant sur l'approche TAP[3], une théorie leur ayant permis de dériver une estimation approximative de LOO pour l'erreur de généralisation. Ainsi, en supposant que l'ensemble des vecteurs de support demeure inchangé durant la procédure LOO, Opper et Winter ont prouvé que :

$$y_p\left(f^0(x_p)-f^p(x_p)\right)=\frac{\alpha_p^0}{(K_{SV}^{-1})_{pp}} \tag{2.8}$$

avec K_{SV} la matrice noyau des vecteurs de support.

Ce qui conduit à :

$$T=\frac{1}{\ell}\sum_{p=1}^{\ell}\Psi(\frac{\alpha_p^0}{(K_{SV}^{-1})_{pp}}-1) \tag{2.9}$$

[3] TAP : Thouless, Anderson et Palmer

2.3.4 Borne de Joachims

Dans [39], Joachims a développé une estimation pour la borne de l'erreur LOO en utilisant les paramètres α_i et les variables d'écart issues de la résolution du problème QP des SVMs. Cette borne s'écrit :

$$T = \frac{1}{\ell} card\left\{i : (2\alpha_i R_\Delta^2 + \xi_i) \geq 1\right\} \tag{2.10}$$

où R_Δ^2 désigne un réel qui satisfait la condition: $c \leq k(x_i, x_j) \leq c + R_\Delta^2$, pour tous les x_i et x_j avec une constante c.

2.3.5 Borne basée sur l'écartement des vecteurs de support

Le concept de l'écartement des vecteurs de support est issu des travaux de Vapnik et Chapelle [40, 41] qui ont proposé l'égalité suivante :

$$y_p\left(f^0(x_p) - f^p(x_p)\right) = \alpha_p^0 S_p^2 \tag{2.11}$$

où S_p désigne la distance entre le point $\phi(x_p)$ et l'ensemble

$$\Lambda_p = \left\{ \sum_{i \neq p, \alpha_i^0 > 0} \lambda_i \phi(x_i), \sum_{i \neq p} \lambda_i = 1 \right\}$$

En utilisant cette estimation, nous avons :

$$T = \frac{1}{\ell} \sum_{p=1}^{\ell} \Psi(\alpha_p^0 S_p^2 - 1) \tag{2.12}$$

2.3.6 Borne "Rayon-Marge"

Cette borne provient des travaux de Vapnik[25]. En considérant la boule englobant les points dans l'espace de noyau, on a :

$$T = \frac{1}{\ell}\frac{R^2}{\rho^2} \tag{2.13}$$

où R désigne respectivement le rayon de la boule englobant les points et ρ la marge du SVM.

Le rayon R est la valeur objective du problème d'optimisation suivant :

$$\begin{aligned} &\min_{a,R} \quad R^2 \\ &\text{sous la contrainte :} \quad R^2 - \left\|\phi(x_i) - a\right\|^2 \geq 0, \quad i = 1,...,\ell \end{aligned} \tag{2.14}$$

Dans [42], il a été montré que cette borne est inappropriée pour faire la sélection de modèle pour les SVM-L1 de marge molle. Une borne modifiée a été alors proposée pour les SVM-L1 en tenant compte des variables d'écart. Cette nouvelle expression s'écrit :

$$T = \frac{1}{\ell}\left[D^2\|w\|^2 + (D^2C+1)\sum_{i=1}^{\ell}\xi_i \right] \tag{2.15}$$

Aussi, dans [4], une borne modifiée "Rayon-Marge" a été proposée et elle se définit par :

$$T = \left(R^2 + \frac{\Delta}{C} \right)\left(\|w\|^2 + 2C\sum_{i=1}^{\ell}\xi_i \right) \tag{2.16}$$

Dans [1], Chapelle et al. ont appliqué la procédure de descente de gradient aux deux dernières bornes de l'erreur de LOO pour sélectionner les hyper-paramètres pour les machines à vecteurs de support. Les résultats expérimentaux obtenus comparés à ceux de la validation croisée classique sont quasiment identiques.

2.4 Erreur empirique

La sélection de modèle pour les machines à vecteurs de support en utilisant l'erreur empirique a été développée par Ayat au cours de sa récente thèse [6]. L'idée derrière cette technique est la minimisation de l'estimation de l'erreur de généralisation à travers une base de validation.

Considérons un problème bi-classe où les observations sont $\{(x_1, y_1),...,(x_\ell, y_\ell)\}$ avec $x_i \in R^d$ et $y_\ell \in \{-1,1\}$. L'hyperplan optimal de séparation entre les deux classes de données est défini par :

$$f(x_i) = \sum_{j=1}^{N_{VS}} \alpha_j y_j k(x_j, x_i) + b \tag{2.17}$$

En posant $t_i = (y_i + 1)/2$, l'étiquette bipolaire des observations devient unipolaire; c'est-à-dire, $t_i = 0$ pour les observations de la classe négative et $t_i = 1$ pour celles de la classe positive. Ainsi, l'erreur empirique pour une observation x_i donnée est définie par l'expression suivante :

$$E_i = |t_i - \hat{p}_i| \tag{2.18}$$

où $\hat{p}_i$ représente un estimé de la probabilité a posteriori associée à l'observation x_i.

L'estimation de la probabilité a posteriori associée à une observation donnée à partir de la sortie brute d'un SVM est réalisée en utilisant la fonction logistique proposée par Platt dans [43]. Cette fonction possède deux paramètres A et B et s'écrit :

$$\hat{p}_i = \frac{1}{1+\exp(A.f_i + B)} \tag{2.19}$$

avec $f_i = f(x_i)$ la sortie brute du SVM.

Les paramètres A et B sont déterminés par une procédure de minimisation de l'entropie croisée, comme présentée par Lin et al.[44] . L'entropie croisée est le critère d'erreur qui provient de la minimisation de –log de la vraisemblance [45] et qui s'écrit :

$$E_{ec} = -\sum_i t_i \log(\hat{p}_i) + (1-t_i)\log(1-\hat{p}_i) \tag{2.20}$$

L'utilisation du modèle développé par Platt pour estimer la probabilité permet de quantifier la distance d'une observation à l'hyperplan déterminé par la machine à vecteurs de support à l'aide d'une fonction continue et dérivable. En effet, l'estimation de probabilité permet de réaliser un calibrage de la distance évaluée par $f(x_i)$ entre 0 et 1 de manière que :

- les observations de la classe positive qui sont bien classées et situées en dehors de la marge aient des probabilités estimées très proches de 1 ;
- les observations de la classe négative qui sont bien classées et situées en dehors de la marge aient des probabilités estimées très proches de 0;
- les observations situées dans la marge aient des probabilités estimées proportionnelles à $f(x_i)$.

Ainsi, avec l'erreur empirique définie, seules les observations mal classées et celles se trouvant dans la marge déterminée par la machine à vecteurs de support sont très importantes puisque les autres observations donnent des erreurs presque nulles. Par conséquent, la minimisation de l'erreur empirique entraîne la réduction des vecteurs de

support (les observations se trouvant dans la marge). En d'autres termes, la minimisation de l'erreur empirique permet de sélectionner des hyper-paramètres définissant une marge contenant moins d'observations; c'est-à-dire une machine avec le moins de vecteurs de support possible; ce qui réduit davantage la complexité du classifieur, confirmation des résultats expérimentaux reportés dans [5].

En effet, nous avons : $E_i = |t_i - \hat{p}_i| = \begin{cases} \hat{p}_i & si \quad y_i = -1 \\ 1 - \hat{p}_i & si \quad y_i = 1 \end{cases}$

Alors $E_i \to 0$ lorsque $\hat{p}_i \to 0$ pour $y_i = -1$ et $\hat{p}_i \to 1$ pour $y_i = 1$

C'est-à-dire $E_i \to 0$ lorsque $f(x_i) < -1$ pour $y_i = -1$ et $f(x_i) > 1$ pour $y_i = 1$

(voir figure 9)

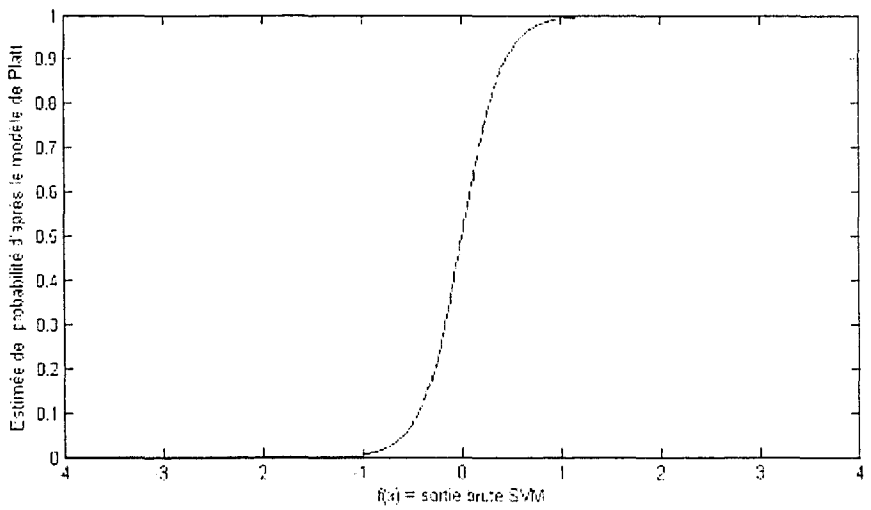


Figure 9 Variation de la probabilité estimée en fonction de la sortie du SVM

Ainsi, la minimisation de l'erreur empirique permet de bien classer les données en dehors de la marge. Ce critère de sélection de modèle est donc utile pour la régularisation de la maximisation de la marge. Entre autre, la réduction de la complexité de la machine avec moins de vecteurs de support permet de réduire la borne de l'erreur en généralisation comme le stipule le théorème 5.2 de Vapnik dans [25] :

L'erreur attendue en généralisation pour une machine à vecteur de support est bornée par : $EP_{error} \leq E \min\left(\frac{N_{VS}}{\ell}, \frac{\left[R^2 |w|^2\right]}{\ell}, \frac{d}{\ell}\right)$.

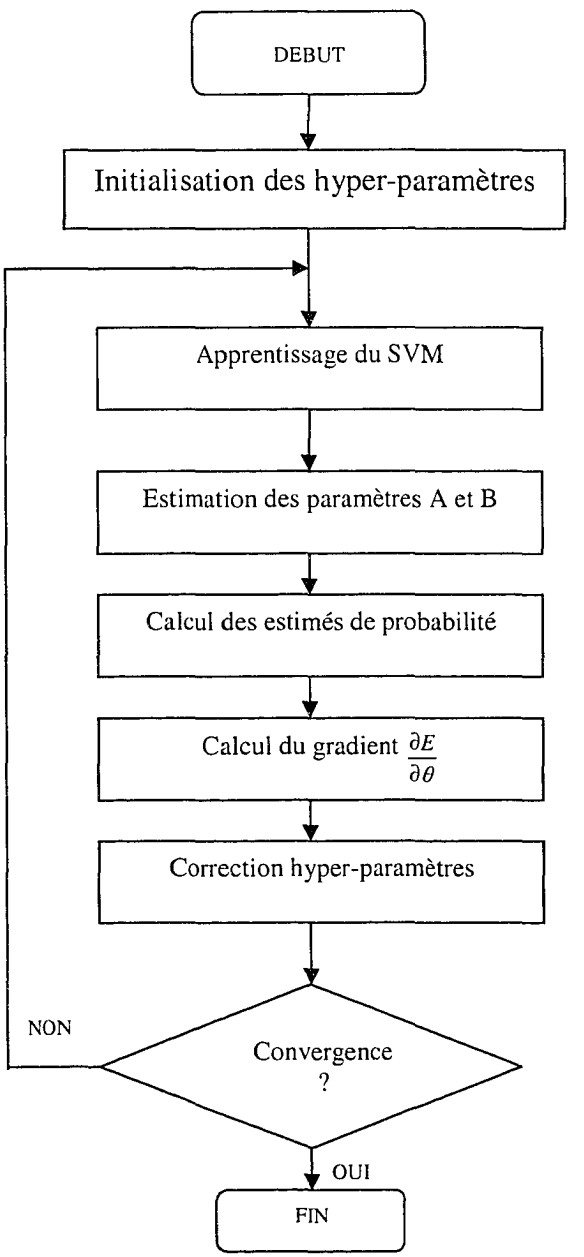


Figure 10 Sélection de modèle basée sur l'erreur empirique

En supposant que la fonction noyau dépend d'un ou plusieurs paramètres, nous notons ces paramètres $\theta = (\theta_1, ..., \theta_n)$. L'optimisation des hyper-paramètres est réalisée par l'algorithme de descente de gradient avec minimisation de $E = \sum E_i$ sur l'ensemble de validation. La figure 10 donne les détails de l'algorithme proposé dans [6].

2.5 Conclusion

Nous avons présenté dans ce chapitre les diverses techniques et critères développés pour faire la sélection de modèle pour les machines à vecteurs de support : le GACV, les bornes d'erreur du LOO et l'erreur empirique. Parmi toutes ces méthodes, se distingue celle de l'erreur empirique dont l'expression est différentiable avec moins de complexité de calcul.

Dans le chapitre suivant, nous allons présenter les deux techniques qui sont : l'approximation du gradient de l'erreur et l'apprentissage incrémental, que nous avons utilisé pour développer l'optimisation des ressources requises pour la sélection de modèle pour les SVM en utilisant l'erreur empirique.

CHAPITRE 3

STRATÉGIES D'OPTIMISATION DES RESSOURCES

3.1 Introduction

La sélection de modèle pour les machines à vecteurs de support basée sur la minimisation de l'erreur empirique a été proposée par Ayat et al.[5] en utilisant une procédure de descente de gradient. Ainsi, à chaque itération, le gradient de l'erreur empirique est calculé après avoir effectué l'apprentissage avec toutes les données disponibles. Dans le but d'optimiser les ressources en matière de temps de CPU et de stockage en mémoire, nous nous sommes intéressés d'une part, à réduire la complexité de calcul du gradient de l'erreur empirique et d'autre part, à utiliser une stratégie d'apprentissage incrémental qui permet de faire évoluer le processus d'incrémentation parallèlement à celui de la descente du gradient.

3.2 Approximation du gradient de l'erreur empirique

Soit N le nombre d'observations constituant l'ensemble de validation, la dérivée de l'erreur empirique est exprimée par :

$$\frac{\partial E}{\partial \theta} = \frac{\partial}{\partial \theta}\left(\frac{1}{N}\sum_{i=1}^{N} E_i\right) = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial E_i}{\partial \theta} \tag{3.1}$$

En se rapportant à la section 2.4, on notera que E_i s'exprime en fonction de l'estimé de la probabilité a posteriori $\hat{p}_i$ qui est une fonction de f_i et la sortie f_i du SVM est à son tour fonction de θ . Ainsi, nous pouvons écrire, en utilisant ces différentes variables intermédiaires :

$$\frac{\partial E_i}{\partial \theta} = \frac{\partial E_i}{\partial \hat{p}_i} \cdot \frac{\partial \hat{p}_i}{\partial f_i} \cdot \frac{\partial f_i}{\partial \theta} \tag{3.2}$$

D'après l'équation 2.18, nous avons :

$$\frac{\partial E_i}{\partial \hat{p}_i} = \frac{\partial \left| t_i - \hat{p}_i \right|}{\partial \hat{p}_i} \tag{3.3}$$

Nous avons des valeurs discrètes pour t_i qui valent 1 pour les données de la classe positive et 0 pour celles de la classe négative. Nous savons aussi que $\hat{p}_i$ est compris entre 0 et 1, donc :

$$\left| t_i - \hat{p}_i \right| = \begin{cases} \hat{p}_i & si \quad t_i = 0 \\ 1 - \hat{p}_i & si \quad t_i = 1 \end{cases}$$

Par conséquent, nous avons :

$$\frac{\partial E_i}{\partial \hat{p}_i} = \begin{cases} 1 & si \quad t_i = 0 \\ -1 & si \quad t_i = 1 \end{cases}$$

Et de façon condensée, nous pouvons écrire :

$$\frac{\partial E_i}{\partial \hat{p}_i} = -y_i \tag{3.4}$$

De la même manière en utilisant l'équation 2.19, donnant l'expression de l'estimé de la probabilité en fonction de f_i, nous avons :

$$\frac{\partial \hat{p}_i}{\partial f_i} = -A\hat{p}_i(1-\hat{p}_i) \tag{3.5}$$

D'après ce qui précède, il reste à évaluer la dérivée partielle $\frac{\partial f_i}{\partial \theta}$ afin de calculer entièrement le gradient de l'erreur empirique. En considérant l'équation 2.17, nous constatons que non seulement les termes $k(x_i, x_j)$ dépendent de θ, mais aussi les termes α_j issus de l'apprentissage du SVM dépendent de θ. Alors la dérivée $\frac{\partial f_i}{\partial \theta}$ s'écrit :

$$\frac{\partial f_i}{\partial \theta} = \sum_{j=1}^{N_{VS}} y_j \left[\frac{\partial k(x_j, x_i)}{\partial \theta} \alpha_j + \frac{\partial \alpha_j}{\partial \theta} k(x_j, x_i) \right] + \frac{\partial b}{\partial \theta} \tag{3.6}$$

Dans l'équation 3.6, l'évaluation de la dérivée $\frac{\partial k(x_i, x_j)}{\partial \theta}$ est aisée et dépend de la nature du noyau choisi. Par contre, celle de la dérivé $\frac{\partial \alpha_j}{\partial \theta}$ est très complexe puisque nous n'avons pas l'expression analytique des termes α_j en fonction de θ. Mais en supposant que tous les vecteurs de support de la machine sont sur la marge, nous pouvons utiliser l'approximation proposée par Chapelle et al. dans [1] :

$$\frac{\partial \alpha}{\partial \theta} = -H^{-1} \frac{\partial H}{\partial \theta} \alpha^T \tag{3.7}$$

où $\alpha = (\alpha_1, ..., \alpha_{N_{VS}}, b)$ et $H = \begin{pmatrix} K^Y & Y \\ Y^T & 0 \end{pmatrix}$

La matrice H de taille $(N_{VS}+1)\times(N_{VS}+1)$ est appelée matrice de Gram modifiée. Ses composantes K_{ij}^{Y} valent $y_i y_j k(x_i, x_j)$ et $Y=(y_1,...,y_{N_{VS}})^T$. D'après l'équation 3.7, nous avons besoin de déterminer l'inverse de la matrice H avant de calculer les dérivées $\frac{\partial \alpha_j}{\partial \theta}$. S'en suit alors une complexité pour le calcul du gradient de l'erreur empirique, puisque nous savons que l'inversion d'une matrice carrée de taille $n\times n$ est de l'ordre de $O(n^3)$ lorsque nous utilisons les algorithmes dérivés de l'élimination gaussienne.

Une analyse expérimentale de la dérivée de $\frac{\partial f_i}{\partial \theta}$ nous a permis de noter que les termes $\frac{\partial \alpha_j}{\partial \theta} k(x_i, x_j)$ sont négligeables par rapport aux termes $\frac{\partial k(x_i, x_j)}{\partial \theta}\alpha_j$. Ainsi, nous pouvons approcher la dérivée de $\frac{\partial f_i}{\partial \theta}$ par :

$$\frac{\partial f_i}{\partial \theta} \approx \sum_{j=1}^{N_{VS}} y_i \frac{\partial k(x_j, x_i)}{\partial \theta}\alpha_j \tag{3.8}$$

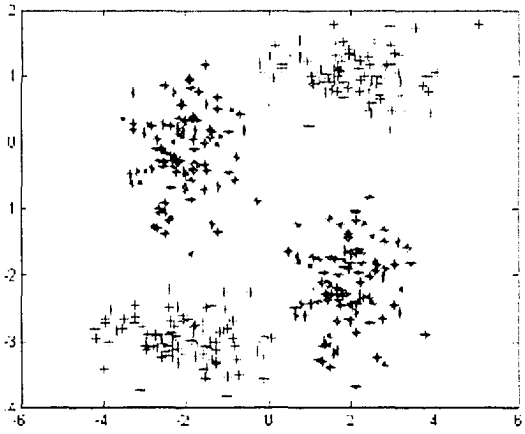
Pour valider l'approximation précédente, nous avons effectué plusieurs tests, tant sur des données synthétiques que sur des données réelles.

Données synthétiques

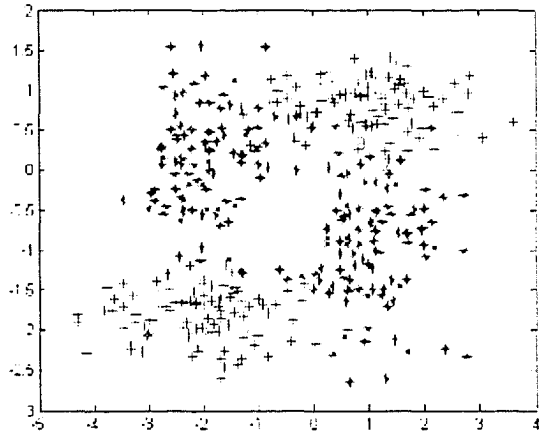
Les problèmes synthétiques que nous considérons représentent des données bi-classe non linéairement séparables. Leurs distributions sont bimodales, problèmes XOR. Les classes sont équiprobables et leurs probabilités conditionnelles sont respectivement

$$p(x \mid y=1)=\frac{1}{2}N(\mu_{11},\Sigma_1)+\frac{1}{2}N(\mu_{12},\Sigma_1) \quad \text{et} \quad p(x \mid y=-1)=\frac{1}{2}N(\mu_{21},\Sigma_2)+\frac{1}{2}N(\mu_{22},\Sigma_2)$$

où $\Sigma_1=\begin{bmatrix}0.5 & 0\\ 0 & 0.5\end{bmatrix}$ et $\Sigma_2=\begin{bmatrix}1 & 0\\ 0 & 0.1\end{bmatrix}$.



(a) Sans chevauchement



(b) Avec chevauchement

Figure 11 Représentation des données du problème XOR

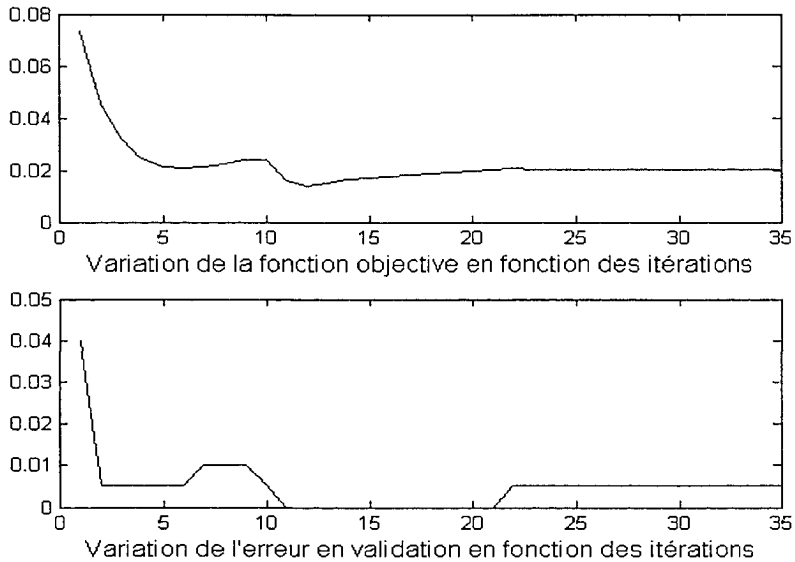
Dans un premier temps, nous considérons deux classes de données sans chevauchement et dans un deuxième temps, des classes avec chevauchement (voir figure 11). Pour contrôler le chevauchement, nous faisons varier la valeur moyenne des distributions normales considérées. Ainsi, pour le cas sans chevauchement, nous prenons $\mu_{11} = (-2,0)^T$, $\mu_{12} = (2,-2)^T$, $\mu_{21} = (2,1)^T$ et $\mu_{22} = (-2,-3)^T$ et pour avoir le chevauchement, nous fixons $\mu_{11} = (-2,0)^T$, $\mu_{12} = (1,-1)^T$, $\mu_{21} = (1,0.8)^T$ et $\mu_{22} = (-2,-1.8)^T$.

Pour chaque problème, la taille de l'ensemble d'apprentissage vaut 400 et celle de l'ensemble de validation 200. La valeur de C est fixé à 10 et nous avons utilisé un noyau gaussien avec initialisation $\gamma = 50$, c'est-à-dire $\sigma^2 = 0.02$.[4]

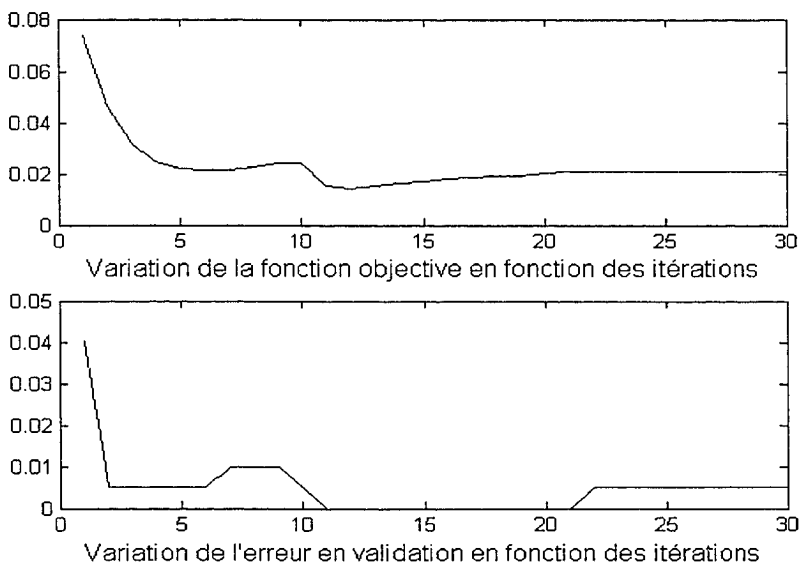
Les figures 12 et 13 montrent les courbes de variation de l'erreur empirique en fonction des itérations et celles du taux de l'erreur en validation au cours de la procédure d'optimisation respectivement pour les deux problèmes XOR décrits ci-dessus. Une analyse des différentes courbes tracées à partir des résultats obtenus permet de conclure d'une part, que la minimisation de l'erreur empirique permet de minimiser le taux d'erreur de reconnaissance sur la base de validation. Et d'autre part, nous pouvons noter que les courbes sont presque les mêmes aussi bien pour le gradient total que pour le gradient approché.

---

[4] Nous utilisons γ au lieu de σ avec $\gamma = \frac{1}{\sigma^2}$ dans l'expression du noyau gaussien

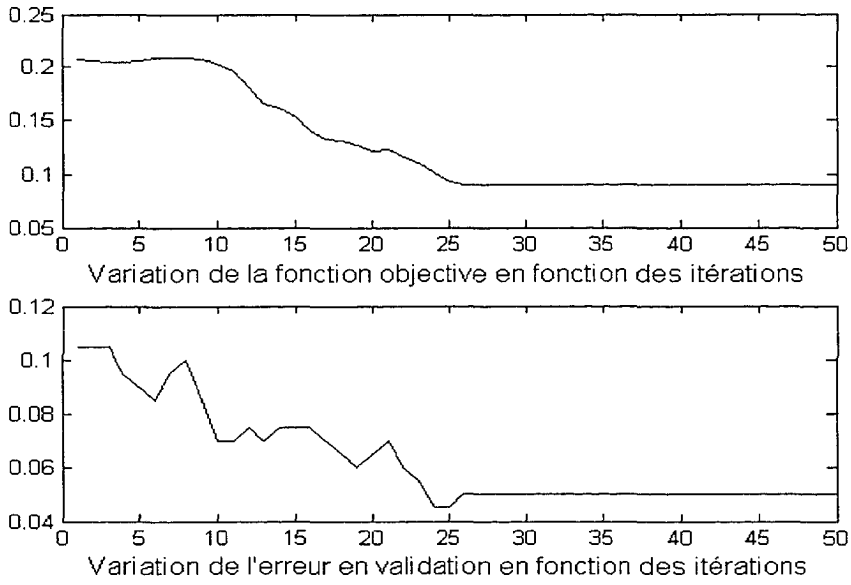


(a) Avec le gradient total (Équation 3.6)

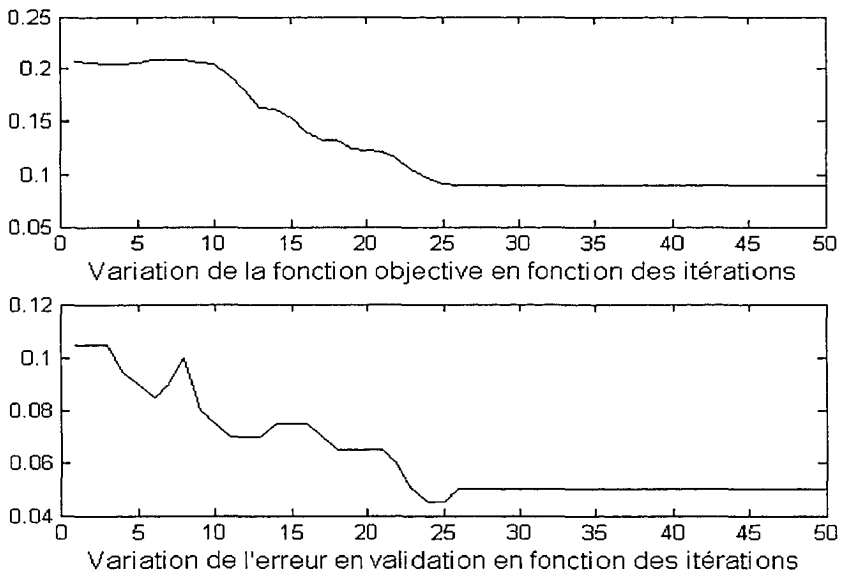


(b) Avec le gradient approché (Équation 3.8)

Figure 12 Variation de l'erreur empirique E et de l'erreur de test en validation au cours de l'optimisation des paramètres du noyau avec les données synthétiques du problème XOR représentées sur la figure 11-a



(a) Avec le gradient total (Équation 3.6)



(b) Avec le gradient approché (Équation 3.8)

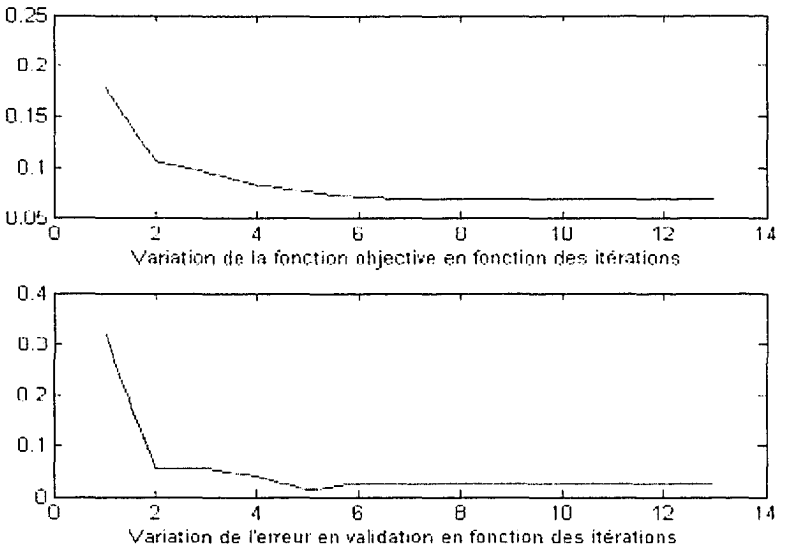
Figure 13 Variation de l'erreur empirique E et de l'erreur de test en validation au cours de l'optimisation des paramètres du noyau avec les données synthétiques du problème XOR représentées sur la figure 11-b

Benchmark UCI

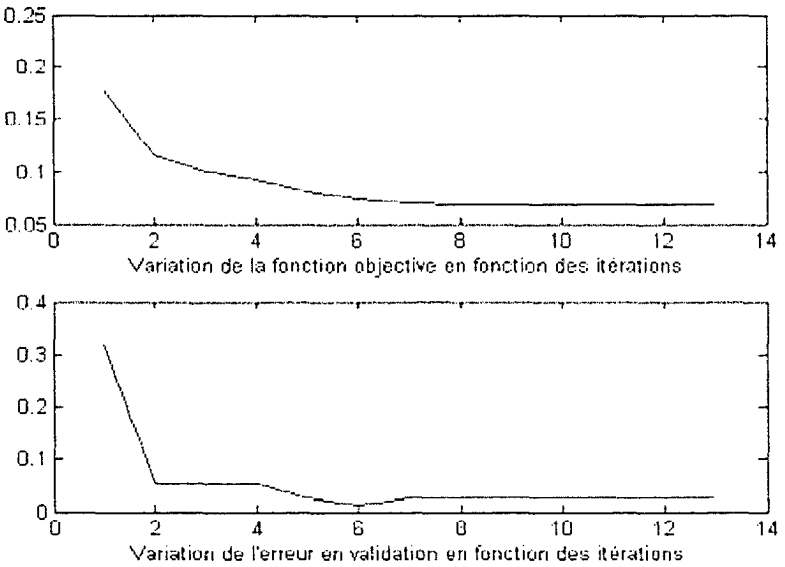
Comme données réelles, nous avons utilisé la base "*thyroid*" issu du benchmark UCI. Cette base de données est constituée d'un problème réel médical. Les caractéristiques sont obtenues à partir des observations et des examens médicaux effectués sur des patients par rapport au fonctionnement de leur thyroïde. La base disponible sur le site de "UCI Machine Learning"[5], décrit un problème bi-classe. Nous avons 100 sous-bases de ce problème. Dans chaque sous-base, l'ensemble d'apprentissage est constitué de 140 observations tandis que la taille de l'ensemble de validation vaut 75. Nous avons utilisé le noyau gaussien comme dans le cas précédent, mais avec initialisation de γ à 12 et nous avons fixé la constante de pénalisation des erreurs C à 1. Les résultats obtenus lors de la minimisation de l'erreur empirique, comme précédemment avec les données synthétiques, sont présentés en figure 14. Nous notons les mêmes conclusions : la performance de la minimisation de l'erreur empirique est identique avec le gradient total et avec le gradient approché.

Ainsi, le gradient de l'erreur empirique peut être évalué en utilisant l'expression de l'équation 3.8 dans l'équation 3.2 donnant la formule de calcul du gradient. Le remplacement du gradient total par le gradient approché permet de nous dispenser de l'inversion de la matrice de Gram modifiée. Par conséquent, le temps de calcul du CPU est considérablement réduit, puisque nous savons que l'inversion d'une matrice est très coûteuse en espace et en temps de calcul.

[5] UCI : University of California, Irvine
http://www.ics.uci.edu/~mlearn/MLRepository.html



a) Avec le gradient total (Équation 3.6)



(b) Avec le gradient approché (Équation 3.8)

Figure 14 Variation de l'erreur empirique E et de l'erreur de test en validation au cours de l'optimisation des paramètres du noyau avec les données *«Thyroïd»* de UCI

3.3 Optimisation des paramètres avec l'apprentissage incrémental

3.3.1 Rappel des techniques d'apprentissage incrémental

L'apprentissage incrémental permet de construire un classifieur de façon séquentielle. C'est une technique qui consiste à caractériser le modèle du classifieur à partir d'une partie des exemples d'apprentissage tout en conservant la capacité de l'améliorer au cours des étapes ultérieures lorsque de nouvelles données d'apprentissage seront disponibles. Ainsi, lors des étapes d'incrémentation, les nouvelles informations contenues dans les nouvelles données sont ajoutées à celles qui étaient mémorisées par le modèle précédemment construit, ce qui permet d'accroître la qualité du classifieur en construction.

La fonction de décision d'une machine à vecteur de support dépend uniquement des exemples se trouvant au voisinage de la marge et non de tous les exemples. Ainsi, pour un ensemble de données d'apprentissage, une machine à vecteur de support résume l'espace des données par une fraction des exemples. Cette qualité des SVM est utilisée pour développer les diverses techniques d'apprentissage incrémental pour les machines à vecteurs de support. Ainsi, à chaque étape d'incrémentation, seuls les vecteurs de support sont retenus pour participer au prochain apprentissage. Cependant, les techniques diffèrent par la formation du nouvel ensemble d'apprentissage pour l'étape suivante. Nous pouvons distinguer quatre diverses techniques rapportées dans [46] :

Technique de partition fixe :

Cette technique a été développée dans [47]. L'ensemble initial d'apprentissage est découpé en sous-ensembles de taille fixe. À chaque itération, les éléments d'un sous-ensemble encore non utilisé sont ajoutés aux vecteurs de support issus de la précédente

étape afin de former le nouvel ensemble d'apprentissage et ainsi de suite jusqu'à épuisement de tous les sous-ensembles.

Technique basée sur l'erreur de classification :

À chaque itération, les nouveaux éléments sont d'abord testés avec le modèle courant. Si un élément est mal classé, il est retenu pour participer au nouvel apprentissage avec les vecteurs de support du modèle courant. Sinon, l'élément est supprimé car on considère que ce dernier n'a pas d'information pertinente pour modifier la frontière de décision. Cette technique est une variante de celle développée dans [48].

Technique basée sur le dépassement de la marge :

Dans ce cas-ci, les nouveaux éléments sont testés pour retenir ceux qui sont à l'intérieur de la marge; c'est-à-dire que tous les nouveaux éléments qui sont en dehors de la marge du modèle de SVM courant sont écartés. Ainsi, le nouvel ensemble d'apprentissage est formé des vecteurs de support en plus des nouvelles données qui sont à l'intérieur de la marge provisoire (mal classés ou pas). Avec cette considération, les éléments se trouvant en dehors de la marge et qui sont mal classés sont considérés comme des données aberrantes et sont ignorées dans les prochaines étapes.

Technique basée sur le dépassement de la marge et l'erreur de classification :

Cette dernière technique permet de prendre en compte les exemples qui sont en dehors de la marge et qui sont mal classés. Ainsi, pour constituer le nouvel ensemble d'apprentissage, on ajoute aux vecteurs de support du modèle courant, les nouveaux exemples mal classés et tous ceux qui sont situés dans la marge et bien classés.

3.3.2 Jumelage de l'apprentissage incrémental avec la sélection de modèle

Le principe des machines «learning» est l'estimation des probabilités a posteriori d'appartenance. Et il est bien connu que la qualité de cette estimation de probabilité est fortement liée à la disposition qualitative et quantitative des données d'apprentissage. Ainsi, le but de généralisation de l'information, que poursuivent les processus d'apprentissage, est plus ou moins atteint en fonction de la taille de l'ensemble d'apprentissage. Cependant, l'impact de la taille de l'ensemble d'apprentissage sur certains classifieurs est moins dramatique que sur d'autres, comme le montre la figure15 tirée de [49]. Ainsi, les machines à vecteurs de support, contrairement à d'autres classifieurs, possèdent un fort pouvoir de généralisation quand la taille de l'ensemble d'apprentissage est petite. Cette propriété des SVM que nous exploitons pour développer notre méthode réside dans le principe de maximisation de la marge lors de la détermination de l'hyperplan de séparation; car cette propriété montre qu'avec peu de données d'apprentissage, la frontière de décision obtenue n'est pas loin du réel. L'erreur empirique estimée est ainsi moins biaisée. Alors au lieu d'utiliser toute la base de données d'apprentissage disponible, surtout quand celle-ci est de taille très importante, nous proposons de faire la sélection de modèle avec un sous-ensemble afin de minimiser le temps de calcul du CPU. Avec la technique d'apprentissage incrémental, nous ajoutons de l'information au modèle à partir des données restantes.

Puisqu'à chaque itération lors de l'optimisation des hyper-paramètres, il faut refaire l'apprentissage du modèle, nous proposons de commencer le processus avec un sous ensemble auquel nous allons ajouter les éléments restants. Le sous ensemble S avec lequel nous débutons le processus d'optimisation est appelé ensemble actif et à chaque itération nous ajoutons une partie ΔS des données restantes. Ainsi, à chaque itération, comme dans le processus normal, le gradient de l'erreur empirique est calculé et utilisé pour faire la mise à jour des hyper-paramètres. De plus, avant de passer à la prochaine itération, nous formons le nouvel ensemble d'apprentissage à partir de certains éléments

de S et une partie des données restantes. En somme, la combinaison de l'apprentissage incrémental et de la procédure de gradient permet d'optimiser simultanément les paramètres α_i et les hyper-paramètres du SVM, ce qui réduit considérablement le temps CPU et l'espace mémoire nécessaire. Les figures 16 et 17 présentent les détails de notre technique.

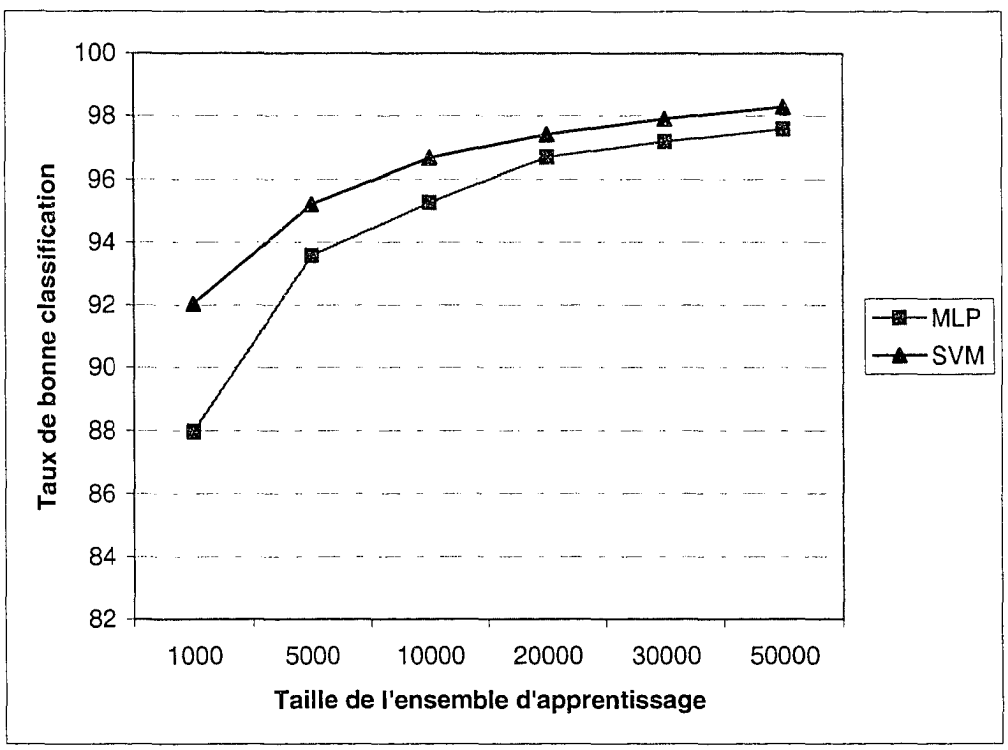


Figure 15 Variation du taux de bonne classification sur l'ensemble de test en fonction de la taille de l'ensemble d'apprentissage (tirée de [49])

1. Initialiser les hyper-paramètres
2. Initialiser l'ensemble actif S
3. Répéter jusqu'à convergence
    3.1. Apprendre le SVM avec l'ensemble S
    3.2. Estimer les paramètres A et B de la sigmoïde
    3.3. Calculer le gradient de l'erreur
    3.4. Corriger les hyper-paramètres
    3.5. Supprimer de S les données éloignées de la marge
    3.6. Ajouter à S une partie ΔS des données restantes

Figure 16 Algorithme d'apprentissage jumelé avec l'optimisation du noyau

Formation de l'ensemble actif

Les diverses techniques utilisées pour former le nouvel ensemble actif sont décrites dans la section précédente où seuls les vecteurs de supports sont préservés pour l'étape incrémentale suivante. Dans notre cas, nous retenons toutes les données qui sont dans et à proximité de la marge parce que les observations qui sont aux abords de la marge peuvent devenir des vecteurs de support. Puisque nous sommes aussi dans un processus d'optimisation des hyper-paramètres, la marge issue du modèle courant de SVM est provisoire et sa variation peut englober plus tard d'autres éléments périphériques. Aux éléments de S retenus, nous ajoutons une partie ΔS des éléments restants dont la taille est choisie dynamiquement en fonction du comportement de la norme du gradient de l'erreur empirique.

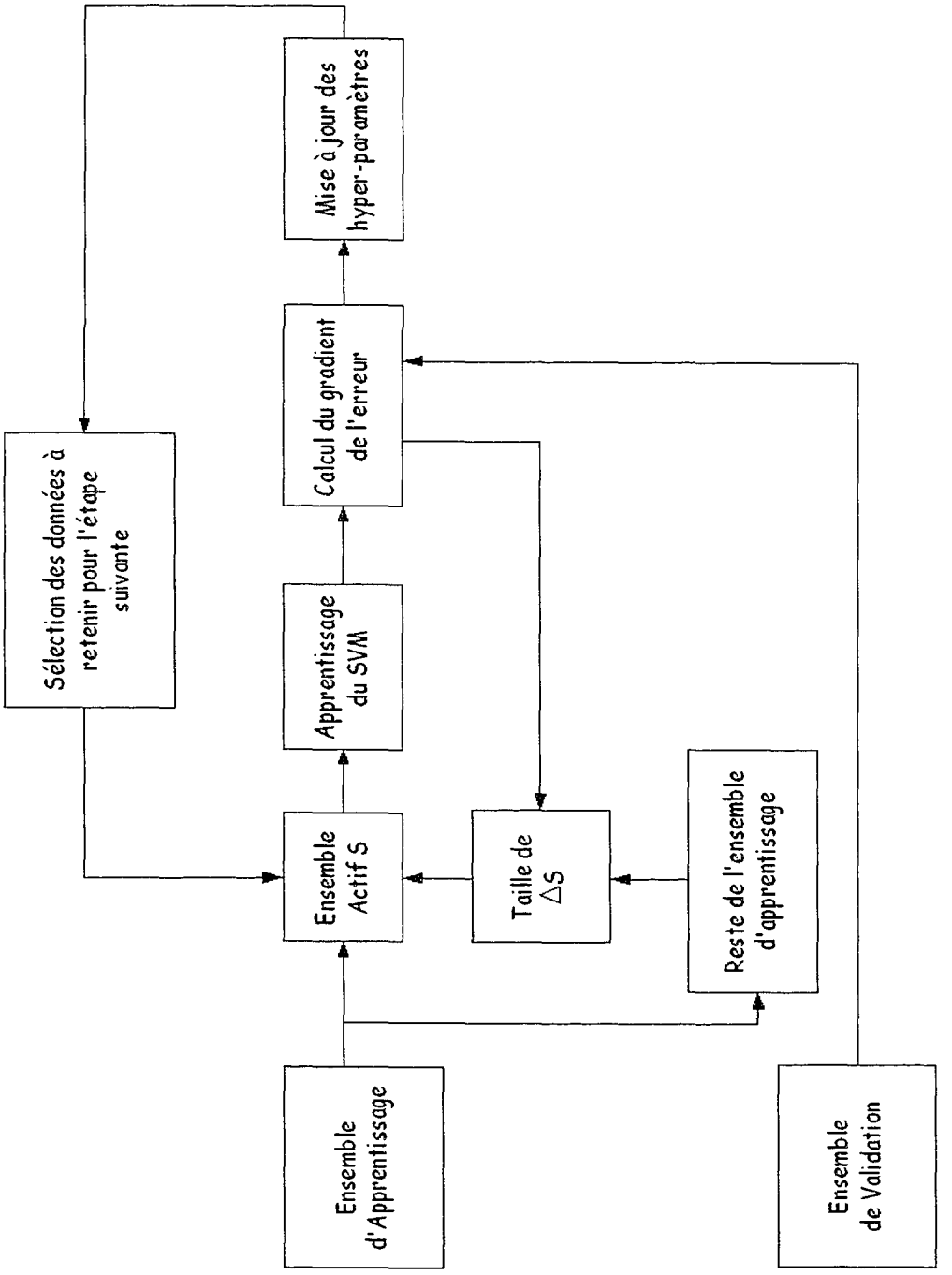


Figure 17 Schéma synoptique du jumelage de l'apprentissage incrémental avec la sélection de modèle

L'idée derrière le choix dynamique de la taille de ΔS est de maintenir la taille de l'ensemble actif S moins importante lorsque la fin du processus d'optimisation des hyper-paramètres n'est pas proche. Ainsi, lorsque la norme du gradient est importante, la taille de ΔS est moins grande, car une forte valeur de la norme du gradient montre que la valeur courante du paramètre est loin de la valeur optimale (voir figure 18). Et lorsque nous tendons vers la convergence, c'est-à-dire la norme du gradient très faible, la taille de ΔS est beaucoup plus grande afin d'améliorer la précision des estimations de probabilités.

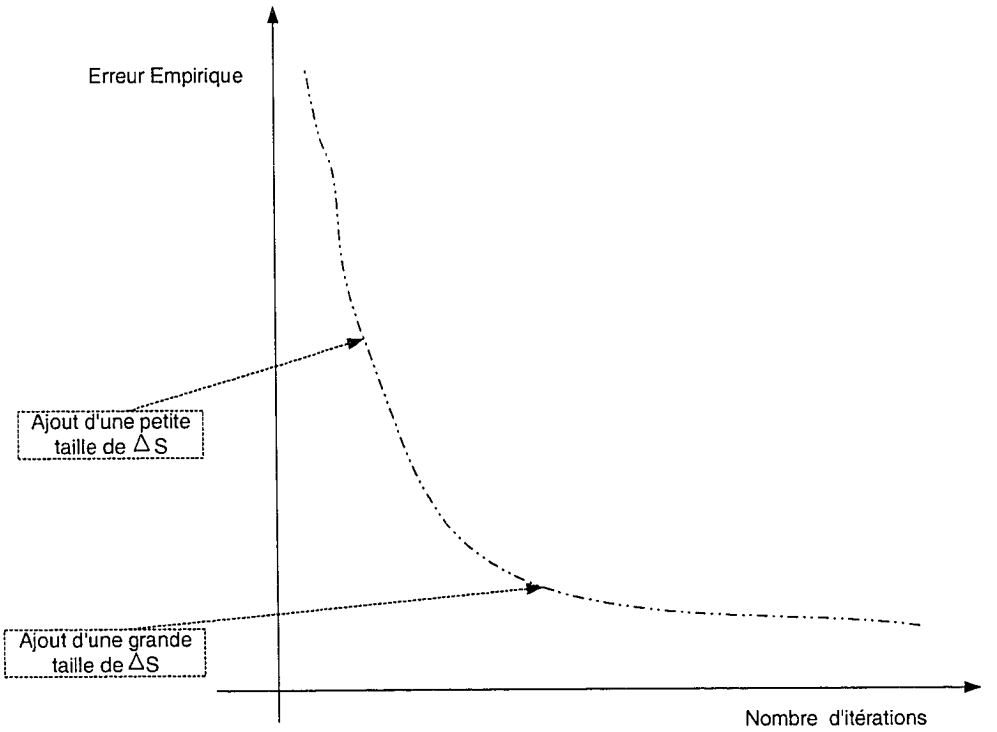


Figure 18 Comportement de la taille de ΔS en fonction de la norme du gradient

3.4 Analyse de l'espace mémoire et de la complexité

Pour faire l'analyse de l'espace mémoire et de la complexité de calcul requise pour l'algorithme proposé, nous introduisons les notations ci-après :

ℓ : taille de l'ensemble d'apprentissage original;

N : taille de l'ensemble de validation;

μ : le facteur représentant le rapport entre la taille de l'ensemble des vecteurs de support et celle de l'ensemble d'apprentissage; [6]

λ : le facteur représentant le nombre de non-vecteurs de support retenus après l'apprentissage au cours du processus incrémental;

β : la fraction de ℓ représentant la taille initiale de l'ensemble actif S;

S_t : la taille de S à la t ème itération;

N_t : le nombre de vecteurs de support à la t ème itération;

N_{VS} : le nombre total de vecteur de support; [7]

r : le nombre de fois que l'ensemble ΔS est ajouté; [8]

T : le nombre total d'itérations pour terminer l'optimisation.

Pour des raisons de simplification, nous supposons que :

- la taille de ΔS est constante et vaut : $\frac{(1-\beta)}{r}\ell$
- après chaque itération, nous retenons $(\mu+\lambda)\%$ des données ayant participé à l'apprentissage
- c_1 est le coût de calcul d'une multiplication

---

[6] Nombre de vecteurs de support proportionnel à la taille de l'ensemble d'apprentissage

[7] En fin de processus $N_t = N_{VS} = \mu\ell$

[8] À partir de la (r+1)ième itération, il n'y a plus de donnée restante à ajouter

3.4.1 Espace mémoire

L'espace mémoire nécessaire pour faire la sélection de modèle est réduite de deux manières en utilisant les deux stratégies proposées.

1. Puisque nous avons approximé l'équation 3.6 par l'équation 3.8, alors le non stockage des matrices H, H^{-1} et $\frac{\partial H}{\partial \theta}$, de taille $(N_{VS}+1)\times(N_{VS}+1)$ chacune, permet de sauver environ $12(N_{VS}+1)^2$ octets à chaque itération[9], en s'inspirant d'une étude similaire d'espace mémoire faite dans [50].

2. La taille réduite de l'ensemble actif S par rapport à la taille totale de l'ensemble d'apprentissage constitue aussi un gain d'espace mémoire, puisque seul l'ensemble actif est chargé en mémoire à chaque itération. Nous rappelons que les données qui sont éloignées de la marge sont supprimées après chaque étape d'apprentissage.

3.4.2 Coût de calcul

L'analyse précise de la complexité de temps de calcul est très difficile, mais nous allons essayer de donner une estimation du coût des opérations effectuées. Ce coût sera évalué en ne considérant que le nombre des opérations de multiplications nécessaires.

Estimation de la taille de l'ensemble des vecteurs de support à chaque itération

Au cours des itérations, le nombre de vecteurs de support varie en proportion des données d'apprentissage ayant déjà participé à l'apprentissage. Au début du processus, après la première itération, nous avons :

[9] La taille d'une variable de type float est 4 octets

$$N_1 = \mu\beta\ell$$

Pour les itérations suivantes, le nombre de vecteurs de support croit linéairement jusqu'à l'étape $t = r+1$, avec les hypothèses posées plus haut :

$$\begin{aligned} N_t &= \mu\beta\ell + \mu\frac{1-\beta}{r}(t-1)\ell \\ &= \left[\beta + \frac{(1-\beta)(t-1)}{r}\right]\mu\ell \\ &\text{si } t \le r+1 \end{aligned} \tag{3.9}$$

Pour toutes les autres itérations $t > r+1$, nous avons $N_t = \mu\ell$

Estimation de la taille de l'ensemble actif S à chaque itération

La taille de l'ensemble actif évolue aussi au cours du processus d'optimisation, par ajout et retranchement de certaines données. Pour la première itération :

$$S_1 = \beta\ell$$

Pour la deuxième itération, nous ne retenons que les données proche de la marge provisoire que nous désignons par : $(\mu+\lambda)\beta\ell$ auquel nous ajoutons $\frac{(1-\beta)}{r}\ell$ données :

$$\begin{aligned} S_2 &= (\mu+\lambda)\beta\ell + \frac{(1-\beta)}{r}\ell \\ &= \left[(\mu+\lambda)\beta + \frac{(1-\beta)}{r}\right]\ell \end{aligned}$$

Pour la troisième itération, nous retenons $(\mu+\lambda)\beta\ell+(\mu+\lambda)\frac{(1-\beta)}{r}\ell$ données auxquelles nous ajoutons $\frac{(1-\beta)}{r}\ell$ des données restantes; ce qui devient :

$$S_3=(\mu+\lambda)\beta\ell+(\mu+\lambda)\frac{(1-\beta)}{r}\ell+\frac{(1-\beta)}{r}\ell$$
$$=\left[(\mu+\lambda)\left(\beta+\frac{(1-\beta)}{r}\right)+\frac{(1-\beta)}{r}\right]\ell$$

Ainsi, de façon générale, à la t-ième itération, nous avons :

$$S_t=\left[(\mu+\lambda)\left(\beta+\frac{(1-\beta)(t-2)}{r}\right)+\frac{(1-\beta)}{r}\right]\ell \text{ si } 2\le t\le r+1$$

Et lorsque $t>r+1$, nous avons :

$S_t=(\mu+\lambda)\ell$ pour toutes les autres itérations jusqu'à $t=T$.

En somme, la taille de l'ensemble actif S vaut :

$$\begin{cases} S_t=\beta\ell \quad si \quad t=1 \\ S_t=\dfrac{(\mu+\lambda)(1-\beta)\ell}{r}(t-2)+\left[(\mu+\lambda)\beta+\dfrac{(1-\beta)}{r}\right]\ell \quad si \quad 2\le t\le r+1 \\ S_t=(\mu+\lambda)\ell \quad si \quad r+2\le t\le T \end{cases} \tag{3.10}$$

Connaissant approximativement la taille de l'ensemble actif et celle de l'ensemble des vecteurs de support au cours du processus d'optimisation, nous allons évaluer le coût du calcul : en premier lieu, le coût de calcul nécessaire pour l'apprentissage et ensuite celui de l'évaluation du gradient.

Coût d'apprentissage

Le coût de l'apprentissage du SVM à chaque itération, d'après la formulation de Dong et al [50], vaut :

$$g_{app}^{t} = \frac{8c_1}{\kappa}(S_t)^2 \tag{3.11}$$

avec κ une constante

Alors, pour toutes les phases d'apprentissage requises pour la sélection de modèle, nous avons :

$$\begin{aligned} g_{app} &= \sum_{t=1}^{T} \frac{8c_1}{\kappa}(S_t)^2 \\ &= \frac{8c_1}{\kappa}\sum_{t=1}^{T}(S_t)^2 \\ &\leq \frac{8c_1}{\kappa}\left(\sum_{t=1}^{T} S_t\right)^2 \end{aligned} \tag{3.12}$$

où

$$\sum_{t=1}^{T} S_t = \ell\left(\beta + \sum_{t=2}^{r+1}[e_1(t-2)+e_2] + \sum_{t=r+2}^{T}(\mu+\lambda)\right)$$

avec $e_1 = \frac{(\mu+\lambda)(1-\beta)}{r}$ et $e_2 = (\mu+\lambda)\beta + \frac{1-\beta}{r}$

Après développement[10], nous retrouvons :

$$g_{app} \leq \frac{8c_1\ell^2}{\kappa}\left(1+(\mu+\lambda)\left[T - \frac{(1-\beta)(r+1)}{2} - 1\right]\right)^2 \tag{3.13}$$

[10] Le détail des calculs se trouve en annexe 1

Une analyse de ce coût nous permet de tirer deux conclusions :

1. Le coût d'apprentissage est inférieur à celui requis pour faire l'apprentissage des machines pour le système proposé initialement par Ayat[6], puisque à chaque itération l'ensemble actif $S_t < \ell$.

2. Le coût peut être davantage réduit avec un bon choix de β et de r; en prenant une petite valeur de β sans pour autant perturber la rapidité de la convergence de l'algorithme et une forte valeur de r proche de T.[11]

Coût de calcul du gradient

Le coût que nous allons évaluer dans cette sous-section ne prend pas en compte l'évaluation des estimées de probabilité et celui de la valeur du noyau et de sa dérivée $\frac{\partial k(x_j, x_i)}{\partial \theta}$.

En considérant les équations 3.2, 3.4, 3.5 et 3.8, le coût du calcul du gradient vaut au cours de la t-ième itération :

$$g^t_{grad} = \sum_{i=1}^{N} (3 + 2N_t)c_1$$

$$g^t_{grad} = Nc_1(3 + 2N_t) \tag{3.14}$$

[11] L'impact du choix de β sera étudié expérimentalement dans la section 5.3.4

En remplaçant l'expression du nombre de vecteurs de support pour chaque itération par l'équation 3.9, nous avons pour toutes les itérations :

$$g_{grad} = Nc_1\left[\sum_{t=1}^{r+1} 2\mu\ell\left(\beta + \frac{(1-\beta)(t-1)}{r}\right) + 2\mu\ell(T-r-1) + 3T\right]$$

Après quelques manipulations algébriques[12], nous avons :

$$g_{grad} = Nc_1\left[3T + 2\mu\ell\left(-\frac{1-\beta}{2}(r+1) + T\right)\right] \tag{3.15}$$

Comparativement au coût du gradient total[13] qui vaut approximativement $NTc_1\left[\tilde{q}_2 N_{VS}^2 + \tilde{q}_1 N_{VS} + \tilde{q}_0\right]$, nous avons :

$$g_{grad} = NTc_1\left[3 + \mu\ell\left(-\frac{1-\beta}{T}(r+1) + 2\right)\right]$$

$$g_{grad} = NTc_1\left[3 + N_{VS}\left(-\frac{1-\beta}{T}(r+1) + 2\right)\right]$$

$$g_{grad} \leq NTc_1\left[\tilde{q}_2 N_{VS}^2 + \tilde{q}_1 N_{VS} + \tilde{q}_0\right]$$

Nous pouvons ainsi conclure que la complexité du calcul du gradient approximé est de l'ordre de $O(N.T.N_{VS})$ tandis qu'elle était de $O(N.T.N_{VS}^2)$ sans l'approximation.

---

[12] Ce calcul est détaillé en annexe 2

[13] Les détails du coût sont en annexe 3 où $\tilde{q}_0$, $\tilde{q}_1$ et $\tilde{q}_2$ sont des coefficients multiplicateurs.

3.5 Conclusion

Dans ce chapitre, nous avons décrit les deux stratégies développées pour optimiser les ressources, le temps de calcul et l'espace mémoire, requises lors de la sélection de modèle pour les SVM en utilisant l'erreur empirique. Nous avons également montré par une étude de la complexité algorithmique que des gains de temps et d'espace sont obtenus à l'aide de ces méthodes. Dans le chapitre 5, nous allons valider toutes ces approches par diverses expérimentations.

CHAPITRE 4

NOUVELLE FORMULATION DU SVM-L1

4.1 Introduction

Les machines à vecteurs de support sont des classifieurs basés sur le principe de la maximisation de la marge provenant de la minimisation du risque structurel. Premièrement développés pour les problèmes linéairement séparables puis pour ceux qui ne le sont pas, les SVM ont connu d'énormes progrès dans des applications diversifiées. Aujourd'hui pour de nombreuses applications, on utilise les SVM à marge molle de norme L1, qui nécessite non seulement le choix des paramètres du noyau mais aussi la valeur du paramètre C. Dans ce chapitre, nous proposons une nouvelle formulation du problème quadratique du SVM-L1, ce qui permet d'inclure le paramètre C dans le choix des paramètres du noyau.

4.2 Description de la nouvelle formulation

L'équation de l'hyperplan de séparation entre les données $\{(x_1, y_1),...,(x_\ell, y_\ell)\}$ d'un problème binaire en utilisant une SVM comme classifieur est donnée par :

$$f(z) = \sum_{i=1}^{\ell} y_i \alpha_i k(x_i, z) + b \qquad (4.1)$$

Dans l'équation 4.1, les réels α_i sont issus de la résolution du problème d'optimisation quadratique décrit par l'équation 4.2.

$$\text{Maximiser} \quad W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j k(x_i . x_j) \tag{4.2}$$

$$\text{Sous les contraintes} \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0 \quad et \quad 0 \le \alpha_i \le C, \quad i = 1, ..., \ell$$

Pour définir la nouvelle formulation du SVM-L1, nous avons utilisé le changement de variable comme proposé dans [4]. Nous posons :

$$\alpha = C\tilde{\alpha} \tag{4.3}$$

Le problème QP donné par l'équation 4.2 devient :

$$\text{Maximiser} \quad W(\tilde{\alpha}) = \sum_{i=1}^{\ell} C\tilde{\alpha}_i - \frac{1}{2} \sum_{i,j=1}^{\ell} C^2 \tilde{\alpha}_i \tilde{\alpha}_j y_i y_j k(x_i . x_j) \tag{4.4}$$

Sous les contraintes

$$\sum_{i=1}^{\ell} C\tilde{\alpha}_i y_i = 0 \tag{4.5}$$

et

$$0 \le C\tilde{\alpha}_i \le C, \quad i = 1, ..., \ell \tag{4.6}$$

L'équation (4.6) permet de rendre le domaine de recherche des coefficients $\tilde{\alpha}_i$ indépendant du paramètre C. Ainsi, nous avons :

$$0 \le \tilde{\alpha}_i \le 1, \quad i = 1, ..., \ell \tag{4.7}$$

De même, la contrainte d'égalité exprimée par l'équation (4.5) devient :

$$\sum_{i=1}^{\ell} C\tilde{\alpha}_i y_i = 0 \quad \Rightarrow \quad C\sum_{i=1}^{\ell} \tilde{\alpha}_i y_i = 0$$

Or, C est un réel positif non nul, nous pouvons alors en déduire l'équation ci-dessous :

$$\sum_{i=1}^{\ell} \tilde{\alpha}_i y_i = 0 \tag{4.8}$$

Considérons maintenant la fonction objective du problème QP :

$$\begin{aligned} W(\tilde{\alpha}) &= \sum_{i=1}^{\ell} C\tilde{\alpha}_i - \frac{1}{2}\sum_{i,j=1}^{\ell} C^2\tilde{\alpha}_i\tilde{\alpha}_j y_i y_j k(x_i.x_j) \\ &= C\sum_{i=1}^{\ell} \tilde{\alpha}_i - \frac{1}{2}C\sum_{i,j=1}^{\ell} C\tilde{\alpha}_i\tilde{\alpha}_j y_i y_j k(x_i.x_j) \\ &= C\left[\sum_{i=1}^{\ell} \tilde{\alpha}_i - \frac{1}{2}\sum_{i,j=1}^{\ell} C\tilde{\alpha}_i\tilde{\alpha}_j y_i y_j k(x_i.x_j)\right] \\ &= C\left[\sum_{i=1}^{\ell} \tilde{\alpha}_i - \frac{1}{2}\sum_{i,j=1}^{\ell} \tilde{\alpha}_i\tilde{\alpha}_j y_i y_j Ck(x_i.x_j)\right] \end{aligned}$$

La stricte positivité du réel C permet de conclure que la maximisation de W par rapport à $\alpha = (\alpha_1,...,\alpha_\ell)$ revient à maximiser $\frac{W}{C}$ par rapport à $\tilde{\alpha} = (\tilde{\alpha}_1,...,\tilde{\alpha}_\ell)$.

Pour compléter notre nouvelle formulation, nous posons $\tilde{k}(x_i.x_j) = Ck(x_i.x_j)$ et nous avons alors :

$$\frac{W(\tilde{\alpha})}{C} = \sum_{i=1}^{l} \tilde{\alpha}_i - \frac{1}{2}\sum_{i,j=1}^{l} \tilde{\alpha}_i\tilde{\alpha}_j y_i y_j \tilde{k}(x_i.x_j) \tag{4.9}$$

Ainsi, nous pouvons reformuler le problème QP de l'équation 4.2 par :

$$\text{Maximiser} \quad W_m(\tilde{\alpha}) = \sum_{i=1}^{l} \tilde{\alpha}_i - \frac{1}{2} \sum_{i,j=1}^{l} \tilde{\alpha}_i \tilde{\alpha}_j y_i y_j \tilde{k}(x_i . x_j) \tag{4.10}$$

$$\text{Sous les contraintes} \quad \sum_{i=1}^{l} \tilde{\alpha}_i y_i = 0 \quad et \quad 0 \leq \tilde{\alpha}_i \leq 1, \quad i = 1, ..., l$$

4.3 Équation de l'hyperplan de séparation avec la nouvelle formulation

L'équation de l'hyperplan de séparation qui permet d'en déduire la fonction de décision lors de la classification est donnée par l'équation 4.1 dans le cas de l'ancienne formulation. En utilisant les changements de variables proposés pour définir la nouvelle formulation, nous avons :

$$\begin{aligned} f(z) &= \sum_{i=1}^{\ell} y_i \alpha_i k(x_i, z) + b \\ &= \sum_{i=1}^{\ell} y_i C \tilde{\alpha}_i k(x_i, z) + b \\ &= \sum_{i=1}^{\ell} y_i \tilde{\alpha}_i C k(x_i, z) + b \end{aligned}$$

Nous en déduisons alors :

$$f(z) = \sum_{i=1}^{\ell} y_i \tilde{\alpha}_i \tilde{k}(x_i, z) + b \tag{4.11}$$

Ainsi, nous n'avons pas besoin de calculer les paramètres α_i avant de définir l'hyperplan, une fois que le problème (4.10) est résolu. Nous venons de montrer par là que le paramètre C qui permet de régulariser la souplesse de la marge peut être incorporé à la définition du noyau.

4.4 Propriétés de $\tilde{k}(x_i.x_j) = Ck(x_i.x_j)$

La fonction $\tilde{k} : (x_i, x_j) \mapsto Ck(x_i.x_j)$ définit une fonction noyau de Mercer si k vérifie le théorème de Mercer à cause de la stricte positivité de C. Cette propriété de la fonction $\tilde{k}$ permet d'affirmer l'existence[14] de solutions pour le nouveau problème QP formulé.

Nous avons :

$$\tilde{k}(x_i.x_j) = Ck(x_i.x_j) = C\phi(x_i).\phi(x_j)$$

En posant $\tilde{k}(x_i.x_j) = \tilde{\phi}(x_i).\tilde{\phi}(x_j)$, nous pouvons en déduire que la fonction de projection implicite utilisée pour projeter les données dans l'espace des caractéristiques dans la nouvelle formulation est donnée par l'expression ci-après :

$$\tilde{\phi}(x_i) = \sqrt{C}\phi(x_i) \tag{4.12}$$

Nous pouvons décomposer l'expression de l'égalité 4.12 comme étant la composition de deux applications:

$$\tilde{\phi}(x_i) = h[\phi(x_i)] \tag{4.13}$$

où h désigne l'homothétie vectorielle de rapport $\sqrt{C}$.

---

[14] Pour s'assurer de l'existence de solution pour un problème quadratique $x^T Qx + r^T x + c$, il faut que la matrice Q soit définie semi-positive, ce qu'on a lorsque la fonction noyau vérifie la condition de Mercer.

Et comme nous le savons, l'application vectorielle homothétie conserve les angles orientés. De plus, faisant partie de la famille des similitudes, l'homothétie conserve les proportions malgré la non conservation des distances. Ainsi, la fonction $\tilde{k}$ issue de l'application $\tilde{\phi}: x_i \mapsto \sqrt{C}\phi(x_i)$, conserve les propriétés de similarité[15] de la fonction noyau mère[16].

Définition de nouveaux noyaux pour la nouvelle formulation

La nouvelle fonction noyau issue du noyau gaussien s'écrit dans la nouvelle formulation, en incorporant l'hyper-paramètre C comme un deuxième hyper-paramètre de la fonction :

$$\tilde{k}(x,y) = C\exp(-\frac{\|x-y\|^2}{\sigma^2}) = a\exp(-b\|x-y\|^2) \tag{4.14}$$

avec a et b des réels positifs.

Mais pour certaines fonctions noyau, l'hyper-paramètre C n'apparaît pas, c'est-à-dire que nous n'avons pas l'apparition d'un hyper-paramètre supplémentaire dans la définition de la fonction noyau. Comme exemple, considérons le noyau polynomial de degré n, où nous disposons déjà de trois hyper-paramètres:

$$k(x,y) = (ax.y + b)^n$$

avec a et b des réels positifs et n un entier .

[15] Par exemple pour les noyaux polynomiaux, la mesure de similarité prend en compte l'orientation des vecteurs.
[16] La fonction k initiale

Alors :

$$\begin{aligned}\tilde{k}(x,y) &= C(ax.y+b)^n \\ &= \left(C^{\frac{1}{n}}\right)^n (ax.y+b)^n \\ &= (C^{\frac{1}{n}}ax.y + C^{\frac{1}{n}}b)^n\end{aligned}$$

D'où :

$$\tilde{k}(x_i,x_j) = (\tilde{a}x.y+\tilde{b})^n \qquad (4.15)$$

avec $\tilde{a} = C^{\frac{1}{n}}a \quad et \quad \tilde{b} = C^{\frac{1}{n}}b$

De la même façon, nous avons aussi pour le noyau KMOD l'introduction de l'hyper-paramètre C dans la définition du noyau sans utilisation d'hyper-paramètre supplémentaire :

$$k(x,y) = a\left(\exp\left(\frac{\gamma^2}{\|x-y\|^2+\sigma^2}\right)-1\right)$$

et

$$\tilde{k}(x,y) = \tilde{a}\left(\exp\left(\frac{\gamma^2}{\|x-y\|^2+\sigma^2}\right)-1\right) \qquad (4.16)$$

avec $\tilde{a} = Ca$

4.5 Avantages de cette nouvelle formulation pour la sélection de modèle

Les différentes méthodes développées pour la sélection de modèle automatique des machines à vecteurs de support utilisent les algorithmes de descente de gradient. Les divers critères proposés sont alors considérés comme des fonctions objectives à optimiser. Mais certains des critères que nous avons rappelés dans le chapitre 2 ne sont pas dérivables par rapport à la variable C. On peut citer comme exemple le critère original basé sur le "Margin-Radius bound" [4]. De plus, certains critères souffrent de problème de convexité de la fonction objective par rapport à la variable C [4].

Les problèmes que nous venons de mentionner dans le précédent paragraphe peuvent être résolus en utilisant cette nouvelle formulation du problème QP, car la dérivabilité du noyau par rapport à C est assurée. Aussi l'hyper-paramètre C, n'étant plus directement lié aux contraintes d'inégalité définissant le problème d'optimisation, n'a plus une influence sur la convexité des fonctions objectives issues des critères de sélection.

Un autre avantage de la nouvelle formulation pour la sélection de modèle est la réduction du nombre d'hyper-paramètres à optimiser. Par exemple, pour les noyaux polynomiaux et KMOD, le nombre d'hyper-paramètres demeure inchangé malgré l'inclusion implicite de C, ce qui rend la sélection de modèle plus aisée tant manuellement que automatiquement.

4.6 Conclusion

Dans ce présent chapitre, nous avons proposé une nouvelle formulation du SVM-L1, ce qui permet de contourner certains problèmes de différentiation lors de l'optimisation de C par de méthodes de descente de gradient. Ainsi, l'hyper-paramètre C peut être considéré et traité comme un paramètre du noyau pour les problèmes de sélection de

modèle pour les SVM. Dans le chapitre 5, nous présenterons les expérimentations et les résultats de cette nouvelle approche dans l'application de sélection de modèle.

CHAPITRE 5

EXPÉRIMENTATIONS ET DISCUSSIONS

5.1 Introduction

Dans cette section, nous présentons les différentes expériences effectuées et les résultats obtenus. Dans un premier temps, nous allons tester nos différentes stratégies sur des données synthétiques et sur des données réelles provenant du répertoire de l'UCI et de MNIST. Les résultats de ces expériences seront comparés à ceux qui sont issus d'autres travaux semblables afin de confirmer la performance de nos stratégies en ce qui concerne la sélection de modèle. Dans un second temps, nous allons examiner certaines propriétés de nos diverses stratégies développées pour optimiser les ressources.

5.2 Performance de nos stratégies dans la sélection de modèle

5.2.1 Données artificielles

Dans cette section, nous allons tester notre méthode[17] sur des données générées artificiellement, appelée *ringnorm*[51]. Ces données sont représentées par des vecteurs de dimension 20 et décrivent un problème à deux classes. La classe 1 est produite par une loi normale multivariée avec une moyenne nulle et une matrice de covariance égale à quatre fois la matrice identité. La classe 2 est aussi produite par une loi normale avec une moyenne $(a,a,...,a)$ et une matrice de covariance égale à la matrice unité. Au cours de notre expérimentation, nous avons utilisé les données disponibles sur le site[18] de l'Université de Toronto où $a = 1/\sqrt{20}$. Nous avons partitionné les 7 400 observations comme suit : 5000 pour l'apprentissage, 1200 pour la validation et le reste, soit 1200, pour la base de test.

---

[17] sélection de modèle avec gradient approché+apprentissage incrémental

[18] http://www.cs.toronto.edu/~delve/data/ringnorm/

L'erreur de Bayes en théorie pour un tel problème est estimée à 1,3%, mais pour les classifieurs testés expérimentalement dont les résultats sont rapportés dans la littérature[2, 51], les taux d'erreur en test sont environ 1,6 à 2,7 %. Nous avons testé notre algorithme sur la base *ringnorm* et nous avons obtenu 1,6% d'erreur en test après convergence du processus avec un noyau RBF. Cette première expérience permet de valider la performance de la minimisation de l'erreur empirique pour la sélection de modèle, ainsi que celle de nos stratégies.

5.2.2 Benchmark UCI

Nous avons utilisé 5 bases de données testées par Chapelle et al[1] provenant du répertoire de données de l'UCI Machine Learning[19]. Chaque base décrivant un problème bi-classe est constituée de 100 différentes sous-bases formées des ensembles d'apprentissage et de test. Le tableau II donne certains détails sur la description de ces ensembles de données.

Tableau II

Description des bases provenant de UCI utilisées

Nom d'identification de la base	Taille de l'ensemble d'apprentissage	Taille de l'ensemble de test	Nombre de caractéristiques
Breast-cancer	200	77	9
Diabetis	468	300	8
Heart	170	100	13
Thyroid	140	75	5
Titanic	150	2051	3

[19] Les données sont disponibles à l'adresse http://ida.first.fhg.de/projects/bench/benchmarks.htm

Nous avons suivi le même protocole expérimental que celui utilisé dans [1] et [2]. Pour chaque base de données, nous procédons à la sélection de modèle en utilisant les cinq premières sous-bases des 100. Puis nous déterminons les valeurs des hyper-paramètres en calculant la valeur médiane des cinq estimations précédemment évaluées. Comme dans [1] et [2], nous avons aussi utilisé le noyau gaussien.

Les résultats obtenus sont montrés dans le tableau III, où nous avons aussi reporté les résultats obtenus avec d'autres différentes techniques de sélection de modèle. Nos résultats sont similaires à ceux issus de la technique de validation croisée [2] ou des méthodes de "Radius-Margin bound" et "Span bound" développées par Chapelle et al[1]. Cependant, le gain en complexité avec notre algorithme est très significatif puisque nous n'inversons pas la matrice H de taille $(N_{VS}+1)\times(N_{VS}+1)$ et de plus nous n'avons pas la résolution d'un problème d'optimisation quadratique supplémentaire.

Tableau III

Taux d'erreur en test trouvés par différents algorithmes de sélection de modèles.

	Cross-validation [2]	Radius margin bound [1]	Span Bound [1]	**Notre Approche [3]**
Breast cancer	26.04±4.74	26.84±4.71	25.59±4.18	25.48±4.38
Diabetis	23.53±1.73	23.25±1.70	23.19±1.67	23.41±1.68
Heart	15.95±3.26	15.92±3.18	16.13±3.11	15.96±3.13
Thyroid	4.80±2.19	4.62±2.03	4.56±1.97	4.70±2.07
Titanic	22.42±1.02	22.88±1.23	22.5±0.88	22.90±1.16

5.2.3 Base de données MNIST

La base de données MNIST provient de la modification de la base NIST contenant les ensembles de données SD-3 (Special Database 3) et SD-1(Special Database 1). Ces ensembles sont des images binaires de chiffres manuscrits isolés, où SD-3 désigne la base d'apprentissage et SD-1 la base de test. L'ensemble SD-3 est constitué à partir d'une collection réalisée chez des employés des bureaux de recensement tandis que SD-1 provient de la collection faite au niveau des lycées et collèges. Ainsi, les ensembles SD-1 et SD-3 ne sont pas de même qualité ; par conséquent les données provenant de SD-3 sont plus claires et faciles à reconnaître que celles de SD-1. C'est ce déséquilibre entre les bases d'apprentissage et de test qui ont amené LeCun et al.[52] à constituer une nouvelle base ("Modified NIST"=MNIST).

L'ensemble SD-1 contient 58 527 images de chiffres manuscrits provenant de 500 personnes différentes. Les auteurs de la base MNIST ont divisé cet ensemble en deux où chaque sous-ensemble contient 30 000 données de 250 personnes. Puis ils ont complété chaque sous-ensemble par des données de SDS-3 de manière à former respectivement l'ensemble d'apprentissage de 60 000 données et l'ensemble de test de 10 000 données. Les images contenues dans la base MNIST ont été normalisées contrairement à celles de la base NIST. Elles sont uniformisées au format 20X20 puis centrées dans une rétine 28X28 en faisant coïncider le centre de gravité du caractère avec le centre géométrique de la rétine. Les images résultantes de ces transformations sont des images à plusieurs niveaux de gris. Pour nos expériences, l'ensemble d'apprentissage est divisé en deux où 50 000 premiers exemples vont servir à l'apprentissage et les 10 000 restants pour la validation.

Nous avons entraîné 45 classifieurs de base, en optant pour la technique "un-contre-un", qui sont couplés pour la décision finale. Les paramètres du noyau de chacun des 45 SVM sont localement optimisés. Nous avons utilisé le noyau RBF en prenant C=10.

Comme pour chaque apprentissage, nous avons environ 5000 exemples de chaque classe, nous initialisons S avec les 2500 premiers exemples et la taille de ΔS des données restantes ajoutées à chaque itération est choisie dynamiquement.

Au cours du test de reconnaissance, nous avons utilisé la règle de décision décrite dans la section 1.6.2 où les fonctions de couplages sont reportées dans le tableau I. Dans le tableau IV, nous présentons les différents résultats selon le couplage utilisé. Ces résultats sont identiques pour les trois algorithmes d'optimisation que nous avons testés, à savoir l'algorithme sans approximation du gradient, celui utilisant la valeur approchée du gradient et celui basé sur l'apprentissage incrémental avec le gradient approché.

Tableau IV

Résultats obtenus avec MNIST en utilisant les différents couplages

Modes de couplage	Taux d'erreur(%)
PWC1	1.6
PWC2	1.5
PWC3	1.7
PWC4	1.6
PWC5	1.5

Au cours de nos expérimentations, nous avons remarqué que les trois algorithmes fournissent à la fin les mêmes valeurs de paramètres de noyau à 10^{-3} près. Ainsi, à la phase de test, nous avons les mêmes taux d'erreur. Cependant, ils se distinguent surtout par le temps de calcul requis. Nous avons noté que la différence de temps de calcul devient très intéressante lorsque la taille de la base des données est importante[20].

[20] Nous en discuterons en détail dans la section 5.3

5.3 Analyse des propriétés de nos stratégies

5.3.1 Impact de l'approximation du gradient

Dans cette section, nous allons étudier expérimentalement l'impact de l'approximation du gradient sur le temps de calcul du CPU. L'expérience réalisée consiste à faire varier la taille de l'ensemble d'apprentissage en faisant exécuter l'algorithme initial[21] d'optimisation de noyau et celui utilisant le gradient approximé. Ensuite, nous procédons à une comparaison des résultats pour analyser le temps de réduction obtenu en approximant le gradient. Pour faire l'expérimentation, nous avons utilisé la base de données MNIST.

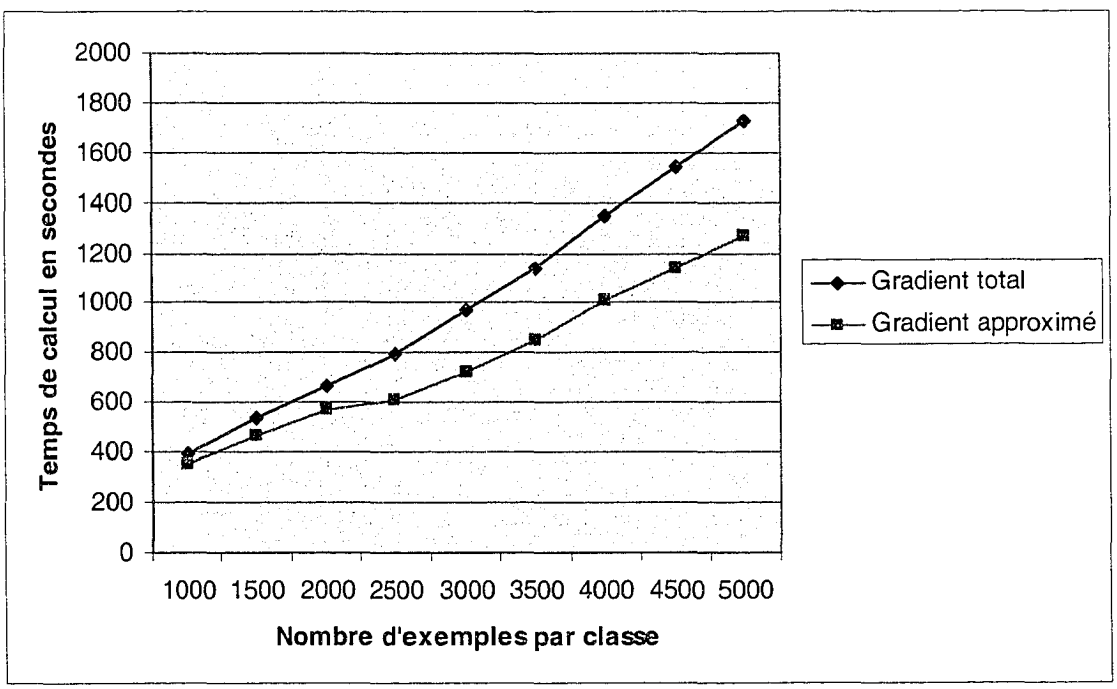


Figure 19 Comparaison entre le temps de calcul requis en fonction de la taille de l'ensemble d'apprentissage (Gradient total vs Gradient approximé)

[21] Algorithme avec le gradient total sans approximation

Les résultats des temps de CPU obtenus sont utilisés pour tracer les courbes de la figure 19 et les diagrammes de la figure 20. Nous pouvons noter sur les deux figures que le temps de réduction s'accroît avec la taille de l'ensemble d'apprentissage. Ainsi, avec le gradient approché, nous réduisons le temps de calcul de 10 à 26%. Ce résultat confirme la théorie, puisque le nombre de vecteurs de support déterminant la taille de la matrice H varie en fonction de l'ensemble d'apprentissage, car plus la taille de H est importante, plus le temps de calcul de son inverse l'est aussi, ce qui engendre une forte réduction de temps.

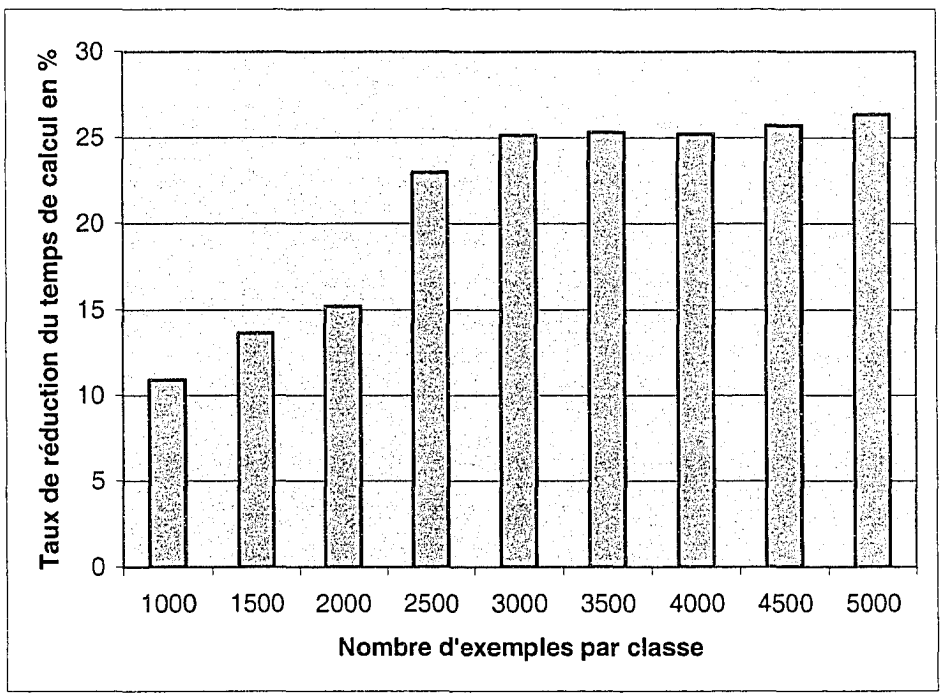


Figure 20 Variation du taux de réduction du temps de calcul en fonction de la taille de l'ensemble d'apprentissage (Gradient total vs Gradient approximé)

5.3.2 Impact de l'apprentissage incrémental

L'apprentissage incrémental permet de débuter l'optimisation des hyper-paramètres et paramètres à partir d'un sous-ensemble de l'ensemble d'apprentissage. Cette technique décrite dans la section 3.3.2 permet de minimiser le temps de calcul sur le total du processus. Nous avons fait une étude comparative comme dans la section précédente pour noter l'influence de cette seule technique sur le temps de calcul. Pour cela, nous avons testé l'algorithme initial et celui avec la technique de l'apprentissage incrémental. Nous rappelons que nous avons utilisé dans les deux algorithmes le gradient total. Les résultats obtenus sont présentés en figures 21 et 22 sous forme de courbes et de diagramme en barre. Nous notons la réduction du temps de 10 à 41% de façon croissante avec la taille totale de l'ensemble d'apprentissage. Nous pouvons aussi remarqué que la réduction du temps de calcul est très significative lorsque la taille de l'ensemble d'apprentissage est importante.

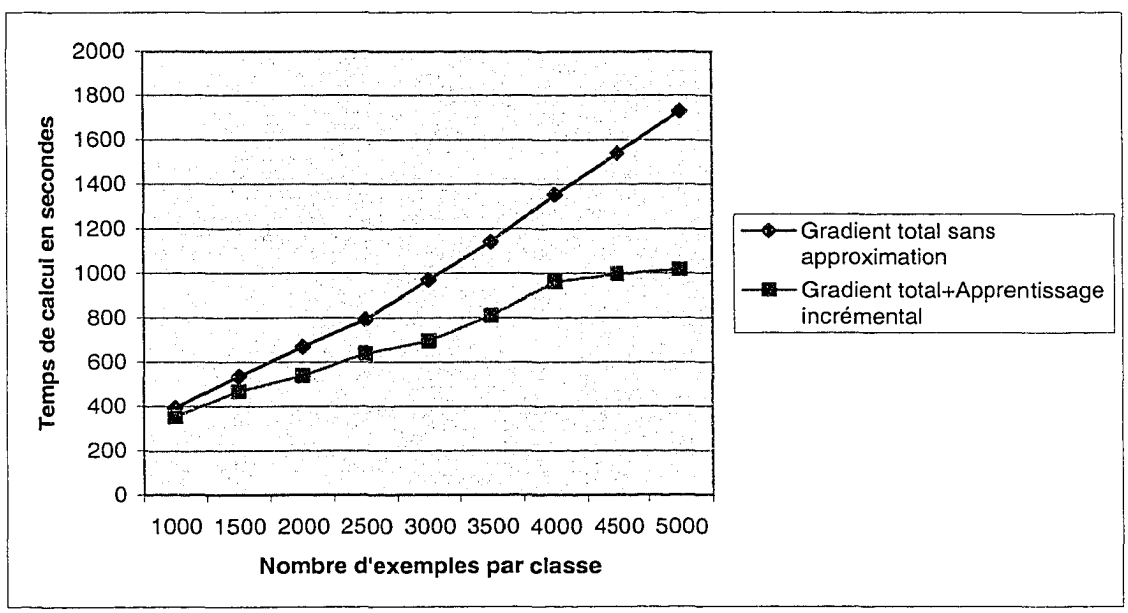


Figure 21 Comparaison entre le temps de calcul requis en fonction de la taille de l'ensemble d'apprentissage (Apprentissage incrémental vs Apprentissage normal)

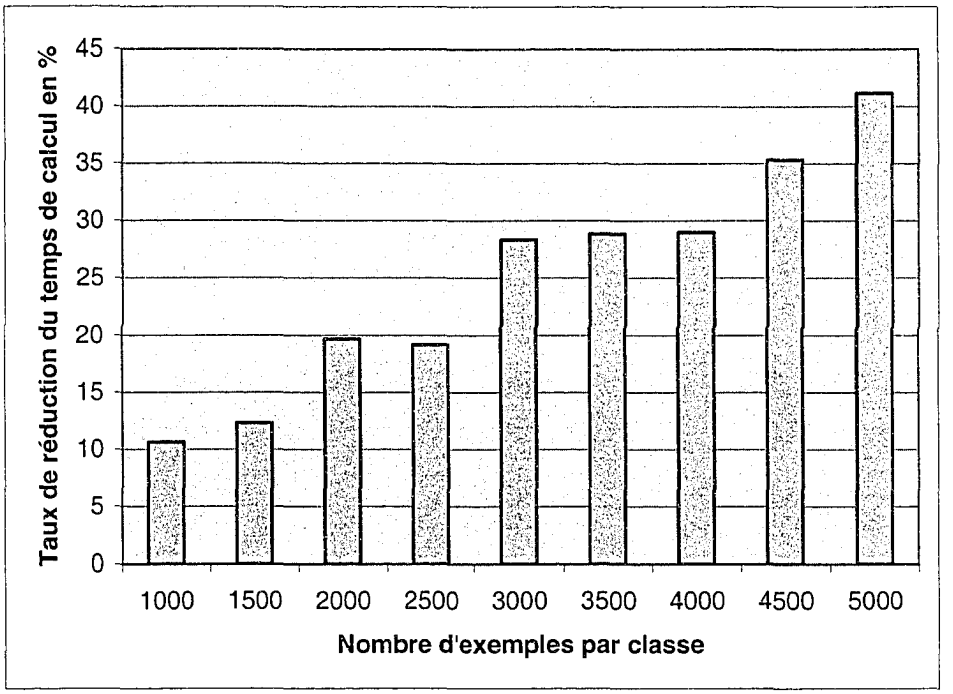


Figure 22 Variation du taux de réduction du temps de calcul en fonction de la taille de l'ensemble d'apprentissage (Apprentissage incrémental vs Apprentissage normal)

5.3.3 Impact global des deux stratégies en fonction de la taille

Nous examinons dans cette section le gain total de temps de calcul de notre modèle par rapport au modèle initial, en combinant les deux stratégies : approximation du gradient et apprentissage incrémental. L'impact global découle des résultats obtenus dans les deux sections précédentes. Cependant, nous n'avons pas une somme systématique des gains obtenus séparément avec les deux stratégies.

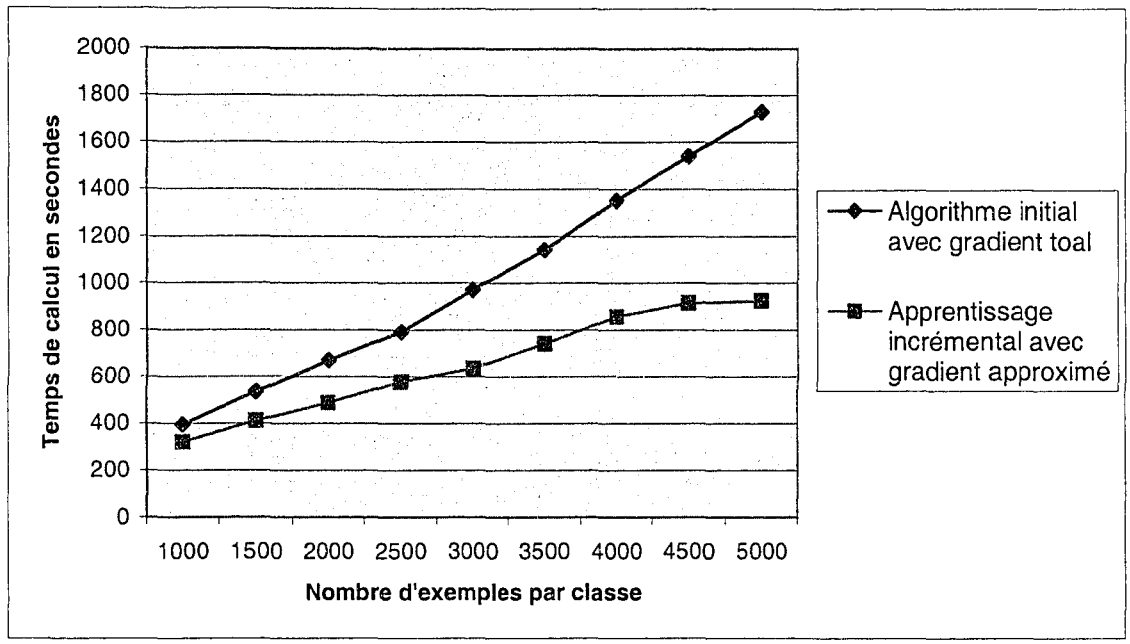


Figure 23 Comparaison entre le temps de calcul requis en fonction de la taille de l'ensemble d'apprentissage

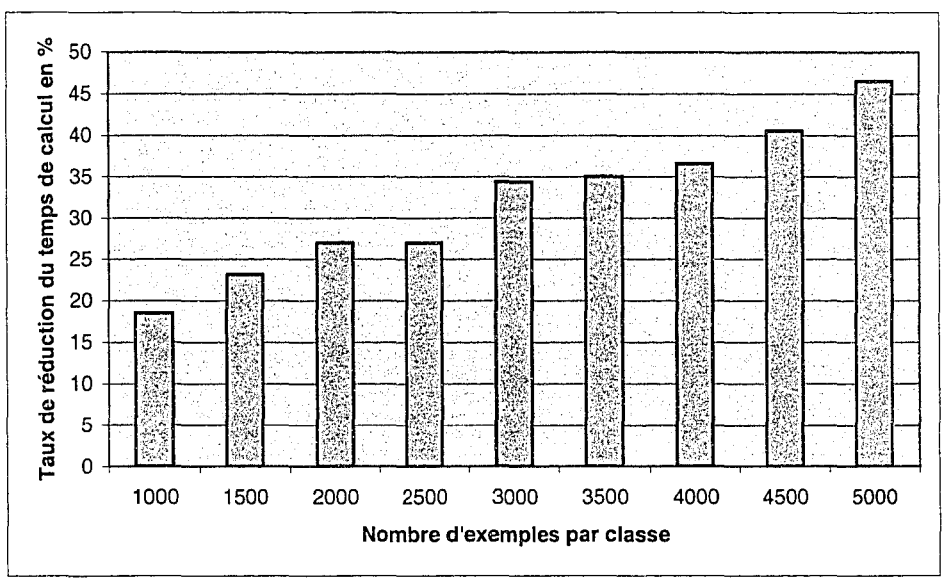


Figure 24 Variation du taux de réduction du temps de calcul en fonction de la taille de l'ensemble d'apprentissage

5.3.4 Impact de la taille initiale de S

Dans la section 3.4, lors de l'analyse de la complexité, nous avons montré qu'un bon choix de la taille initiale[22] de S a une influence sur le temps d'apprentissage total pour tout le processus d'optimisation; par conséquent, sur le temps de calcul final. Nous avons testé les données des classes de chiffres 0 et 1 en formant un problème bi-classe. Nous avons fait varier la taille d'initialisation de l'ensemble actif S et après la convergence du processus, nous notons le temps de calcul. Nous avons considéré que la taille totale de l'ensemble d'apprentissage vaut 6000, 8000 ou 10000. Dans chaque cas, nous avons supposé que la taille initiale de S est successivement 20%, 30%, 40%, 50%, 60% ou 70% de la taille totale. La figure 25 montre les résultats obtenus sous forme de courbes où la conclusion 2 de la section 3.4.2 est confirmée, soit qu'une petite valeur de β permet de réduire le temps de calcul.

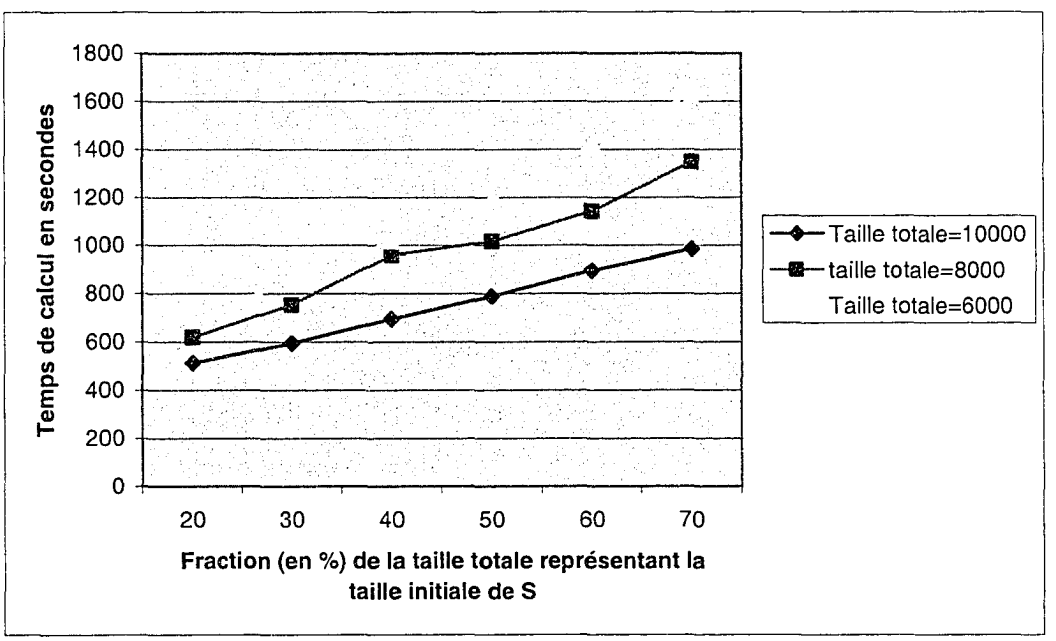


Figure 25 Courbes montrant l'impact de la taille initiale de S

[22] le choix de β

5.4 Expérimentation de la nouvelle formulation du SVM-L1

Dans cette section, nous allons effectuer des expérimentations avec la nouvelle formulation du SVM-L1 développée en chapitre 4. Nous allons d'abord tester les données artificielles utilisées dans la section 5.2.1, puis les cinq bases de données de UCI utilisées dans la section 5.2.2 et enfin la base MNIST.

Nous avons réalisé la sélection de modèle sur ces bases d'apprentissage en utilisant les algorithmes décrivant la nouvelle formulation avec le critère de l'erreur empirique. Nous avons utilisé le noyau gaussien modifié $\tilde{k}(x,y) = a\exp(-b\|x-y\|^2)$ avec deux paramètres *(a,b)* comme nous l'avons proposé dans le chapitre 4. Ces deux paramètres sont optimisés simultanément par une procédure de descente de gradient.

Pour les données artificielles, après convergence de la procédure de sélection, nous avons obtenu a=1.5246 et b=0.0268. Avec ces paramètres du noyau optimisé, le taux de reconnaissance en test vaut 1,5% contre 1,6% obtenu dans la section 5.2.1 où l'hyper-paramètre C était fixe et seule la variance du noyau gaussien était optimisée.

Quant aux données de la base UCI, les résultats que nous avons obtenus sont presque identiques à ceux obtenus dans la section 4.2.2. Nous avons de légères améliorations sur certaines bases et de légères dégradations sur d'autres. Nous pouvons dire que les améliorations proviennent de l'optimisation simultanée des deux hyper-paramètres. Cependant, pour les dégradations, il pourrait s'agir d'un problème de minimum local. Pour les expérimentations de la section 4.2.2, l'hyper-paramètre C était fixe et seule la variable γ du noyau RBF est optimisée, ce qui n'est pas le cas pour les expériences de la présente section qui sont plus sensibles au problème de minimum local, puisque nous avons deux paramètres à optimiser.

Tableau V

Résultats obtenus avec la nouvelle formulation sur les cinq baes de UCI en comparaison avec les résultats testés en section 5.2.2

	QP normal			QP modifié		
	C Fixe*	γ optimisé	Erreur en test	$a\,(C)$ optimisé	$b\,(\gamma)$ optimisé	Erreur en test
B-cancer	1.00	0.0431	25.48±4.38	1.080	0.100	25.79±4.21
Diabetis	0.50	0.0580	23.41±1.68	0.500	0.060	23.25±1.82
Heart	0.50	0.0393	15.96±3.13	0.666	0.010	15.98±3.32
Thyroid	1.00	0.2911	4.70±2.07	10.00	0.183	4.64±2.13
Titanic	1.00	0.2027	22.90±1.16	1.100	0.1197	22.93±1.17

* La valeur de C est fixée de façon empirique après avoir testé certaines valeurs.

Pour le problème multi-classe avec la base MNIST, nous avons optimisé les hyper-paramètres a et b pour les 45 classifieurs SVM. À la convergence du processus, nous avons noté que les valeurs de b (γ) sont relativement de même ordre que les valeurs trouvées dans la section 5.2.3. Mais en ce qui concerne les valeurs de a=C, nous avons remarqué que la valeur trouvée traduit la nature du problème bi-classe qui a été traité en terme de la séparabilité : une grande valeur pour les problèmes fortement séparables et une petite valeur pour ceux qui ne le sont pas.

Tableau VI

Tableau donnant le nombre d'éléments mal classés en test (bi-classe) par la nouvelle formulation vs par l'ancienne formulation

	Cl.2	Cl.3	Cl.4	Cl.5	Cl.6	Cl.7	Cl.8	Cl.9	Cl.10
Cl. 1	0	0	0	0	0	1	0	2	0
Cl. 2		0	-1	0	0	1	0	0	0
Cl. 3			0	0	0	0	0	0	-1
Cl. 4				0	-1	0	0	-1	0
Cl. 5					0	1	0	0	0
Cl. 6						0	0	-1	0
Cl. 7							0	0	0
Cl. 8								1	0
Cl. 9									0

* une valeur positive = la nouvelle formulation a mieux classé ce nombre d'éléments plus que l'ancienne

* une valeur négative = la nouvelle formulation a mal classé ce nombre d'éléments plus que l'ancienne

En test, nous avons eu les mêmes résultats comme dans la section 5.2.3 à l'exception du couplage PWC3 qui s'est amélioré de 0.1% . Nous nous attendions normalement à une amélioration générale de la capacité de généralisation des classifieurs. Et pour y voir clair, nous avons comparé le taux d'erreur de chacun des 45 classifieurs en considérant un problème bi-classe. Les taux d'erreur obtenus sont mentionnés dans les tableaux de l'annexe 4. Le tableau VI regroupe les nombres d'observations de test qui ont été mieux classées avec la nouvelle formulation. Dans ce tableau, nous pouvons remarquer une

amélioration pour 5 classifieurs et une légère dégradation pour 5 autres, tandis que la capacité des 35 restants est inchangée. Une explication à ces observations pourrait être attribuée au problème de minimum local comme nous l'avons expliqué pour les bases de UCI. Cette observation retient beaucoup notre attention et demande de mettre plus d'effort pour une profonde explication.

5.5 Conclusion

Dans ce chapitre, nous avons reporté les différentes expérimentations effectuées pour confirmer les stratégies d'optimisation développées dans le chapitre 3. Nous avons réalisé des expériences tant avec des données artificielles qu'avec des données réelles. Et les résultats obtenus dans les deux cas permettent de confirmer les diverses approches proposées. Nous avons aussi fait des tests pour décrire certaines propriétés de l'algorithme proposé.

Nous avons également testé la nouvelle formulation des SVM-L1 que nous avons développée dans le chapitre 4. Nous avons appliqué cette nouvelle formulation pour des problèmes de sélection de modèle et nous avons eu des résultats encourageants. Nous pensons aussi développer cette nouvelle formulation pour d'autres applications.

En somme, ce dernier chapitre de notre mémoire a permis de valider les diverses approches que nous avons développées au cours de nos recherches.

CONCLUSION GÉNÉRALE

Les machines à vecteurs de support sont des classifieurs qui ont une excellente capacité de généralisation lorsque les hyper-paramètres sont adéquatement choisis. Cependant, les diverses méthodes développées pour optimiser ces hyper-paramètres de façon automatique sont très coûteuses en temps de calcul et en espace mémoire, ce qui restreint l'application de ces méthodes à des problèmes réels possédant de large base de données. Dans ce mémoire, nous avons présenté deux techniques qui permettent de faire la sélection de modèle efficacement avec réduction de ressources en terme de temps de calcul et de l'espace mémoire. Ces techniques développées pour l'optimisation des ressources sont :

- La *technique du gradient approché* qui permet de réduire la complexité de calcul du gradient de l'erreur empirique. Cette procédure sert à évaluer le gradient sans avoir recours à l'inversion de la matrice Grahm des vecteurs de support.
- la technique qui sert à jumeler l'apprentissage des hyper-paramètres et des paramètres des machines. Ainsi, la sélection de modèle et l'apprentissage de la machine sont réalisés simultanément.

Pour valider notre approche, nous avons testé ces différentes stratégies d'optimisation sur des données synthétiques, sur les bases de données de UCI et sur un problème réel de reconnaissance de chiffres manuscrits. Les expérimentations effectuées ont ainsi permis de montrer de façon pratique les gains de temps de calcul et d'espace mémoire. Nous avons aussi étudié certaines propriétés de l'algorithme d'optimisation que nous avons développé et nous avons remarqué que le gain en temps croît en fonction de la taille de l'ensemble d'apprentissage. Ainsi, pour des problèmes avec de large base de données, nos stratégies sont très efficaces pour réaliser une bonne sélection de modèle pour les machines à vecteurs de support avec moins de ressources.

Outre ces deux techniques développées, nous avons aussi proposé dans ce mémoire une nouvelle formulation pour les machines à vecteurs de support de norme L1. Une formulation qui permet d'inclure l'hyper-paramètre C dans la définition de la fonction noyau, ce qui permet de ramener le problème de sélection de modèle au seul problème d'optimisation des paramètres du noyau. Cette nouvelle formulation a été aussi testé et a donné de résultats encourageants. Et pour des recherches futures, nous pensons étendre cette nouvelle formulation pour les SVM-L1 aux autres critères de sélection de modèle, surtout à ceux qui souffrent des problèmes de différentiation de premier ou de second ordre. Nous nous intéresserons aussi à une étude plus approfondie pour donner une explication plus fournie quant aux petites dégradations remarquées lors de l'expérimentation de la nouvelle formulation sur certaines données.

ANNEXE 1

Détails du calcul du coût d'apprentissage

$$g_{app} \le \frac{8c_1}{\kappa}\left(\sum_{t=1}^{T} S_t\right)^2 = \frac{8c_1\ell^2}{\kappa}\left(\beta + \sum_{t=2}^{r+1}\left[e_1(t-2)+e_2\right] + \sum_{t=r+2}^{T}(\mu+\lambda)\right)^2$$

$$= \frac{8c_1\ell^2}{\kappa}\left(\beta + \sum_{t=0}^{r-1}\left[e_1 t + e_2\right] + \sum_{t=r+2}^{T}(\mu+\lambda)\right)^2$$

$$= \frac{8c_1\ell^2}{\kappa}\left(\beta + e_2 + \sum_{t=1}^{r-1}\left[e_1 t + e_2\right] + \sum_{t=r+2}^{T}(\mu+\lambda)\right)^2$$

$$= \frac{8c_1\ell^2}{\kappa}\left(\beta + e_2 + \sum_{t=1}^{r-1} e_1 t + \sum_{t=1}^{r-1} e_2 + \sum_{t=r+2}^{T}(\mu+\lambda)\right)^2$$

$$= \frac{8c_1\ell^2}{\kappa}\left(\beta + e_2 r + e_1\sum_{t=1}^{r-1} t + \sum_{t=r+2}^{T}(\mu+\lambda)\right)^2$$

$$= \frac{8c_1\ell^2}{\kappa}\left(\beta + e_2 r + e_1\frac{r(r-1)}{2} + (\mu+\lambda)(T-r-1)\right)^2$$

$$= \frac{8c_1\ell^2}{\kappa}\left(1 + \beta r(\mu+\lambda) + \frac{(\mu+\lambda)(1-\beta)(r-1)}{2} + (\mu+\lambda)(T-r-1)\right)^2$$

$$= \frac{8c_1\ell^2}{\kappa}\left(1 + (\mu+\lambda)\left[\beta r + \frac{(1-\beta)(r-1)}{2} + T - r - 1\right]\right)^2$$

$$= \frac{8c_1\ell^2}{\kappa}\left(1 + (\mu+\lambda)\left[-\frac{(1-\beta)(r+1)}{2} + T - 1\right]\right)^2$$

ANNEXE 2

Calcul détaillé du coût d'estimation du gradient approché

$$g_{grad} = Nc_1\left[\sum_{t=1}^{r+1} 2\mu\ell\left(\beta + \frac{(1-\beta)(t-1)}{r}\right) + 2\mu\ell(T-r-1) + 3T\right]$$

$$g_{grad} = Nc_1\left[2\mu\ell\beta(r+1) + 2\mu\ell\sum_{t=1}^{r+1}\left(\frac{(1-\beta)(t-1)}{r}\right) + 2\mu\ell(T-r-1) + 3T\right]$$

$$g_{grad} = Nc_1\left[2\mu\ell\frac{(1-\beta)}{r}\sum_{t=1}^{r+1}(t-1) + 2\mu\ell(\beta r + \beta + T - r - 1) + 3T\right]$$

$$g_{grad} = Nc_1\left[2\mu\ell\frac{(1-\beta)}{r}\sum_{t=0}^{r} t + 2\mu\ell(\beta r + \beta + T - r - 1) + 3T\right]$$

$$g_{grad} = Nc_1\left[2\mu\ell\frac{(1-\beta)}{r}.\frac{r(r+1)}{2} + 2\mu\ell(\beta r + \beta + T - r - 1) + 3T\right]$$

$$g_{grad} = Nc_1\left[2\mu\ell\left(\frac{(1-\beta)(r+1)}{2} + \beta(r+1) + T - (r+1)\right) + 3T\right]$$

$$g_{grad} = Nc_1\left[2\mu\ell\left(T + (r+1)\left(\frac{(1-\beta)}{2} + \beta - 1\right)\right) + 3T\right]$$

$$g_{grad} = Nc_1\left[2\mu\ell\left(T + (r+1)\frac{\beta-1}{2}\right) + 3T\right]$$

ANNEXE 3

Calcul détaillé du coût d'estimation du gradient total

Pour chaque donnée de validation, le gradient de l'erreur empirique s'écrit :

$$\frac{\partial E_i}{\partial \theta} = A y_i \hat{p}_i (1-\hat{p}_i) \sum_{j=1}^{N_{VS}} y_j \left[\frac{\partial k(x_j, x_i)}{\partial \theta} \alpha_j + \frac{\partial \alpha_j}{\partial \theta} k(x_j, x_i) \right] + \frac{\partial b}{\partial \theta}$$

$$\text{avec } \frac{\partial \alpha}{\partial \theta} = \left(\frac{\partial \alpha_1}{\partial \theta}, ..., \frac{\partial \alpha_{N_{VS}}}{\partial \theta}, \frac{\partial b}{\partial \theta} \right) = -H^{-1} \frac{\partial H}{\partial \theta} \alpha^T$$

Pour évaluer H^{-1} avec une technique basée sur l'élimination gausienne, il faut approximativement $q(N_{VS}+1)^3$ multiplications, avec q un coefficient multiplicateur. Ainsi, le coût de calcul de $\frac{\partial \alpha}{\partial \theta}$ vaut :

$$q(N_{VS}+1)^3 + (N_{VS}+1)^3 + (N_{VS}+1)^2 = (q+1)(N_{VS}+1)^3 + (N_{VS}+1)^2$$

Le calcul de $\frac{\partial \alpha}{\partial \theta}$ est fait une seule fois par itération, donc en moyenne nous avons pour chaque observation de validation un coût de :

$\frac{(q+1)(N_{VS}+1)^3 + (N_{VS}+1)^2}{N} \approx (q+1)(N_{VS}+1)^2 + (N_{VS}+1)$ si nous supposons que le nombre de vecteurs de support est sensiblement égal aux nombre d'observations contenues dans la base de validation.

Par suite, le coût d'évaluation de $\frac{\partial E_i}{\partial \theta}$ vaut :

$$(q+1)(N_{VS}+1)^2 + (N_{VS}+1) + 4 + 3N_{VS} = \tilde{q}_2 N_{VS}^2 + \tilde{q}_1 N_{VS} + \tilde{q}_0$$

avec $\tilde{q}_2 = q+1$, $\tilde{q}_1 = 2q+6$ et $\tilde{q}_0 = q+6$.

ANNEXE 4

Comparaison des taux d'erreur par problème biclasse entre la nouvelle et l'ancienne formulation

Taux d'erreur par problème bi-classe en utilisant la nouvelle formulation

	Cl.1	Cl.2	Cl.3	Cl.4	Cl.5	Cl.6	Cl.7	Cl.8	Cl.9	Cl.10
Cl.1	0	0.0473	0.4970	0.0503	0.0510	0.4274	0.3612	0.0996	0.3582	0.3017
Cl.2	0	0	0.2307	0.0932	0.0945	0.0493	0.0956	0.2312	0.0474	0.1399
Cl.3	0	0	0	0.2449	0.3476	0.1040	0.4020	0.8252	0.3988	0.4410
Cl.4	0	0	0	0	0.1004	0.6309	0.0508	0.3925	0.4536	0.5944
Cl.5	0	0	0	0	0	0.0534	0.3608	0.2488	0.2556	0.7534
Cl.6	0	0	0	0	0	0	0.9189	0.2083	0.3751	0.4734
Cl.7	0	0	0	0	0	0	0	0.1007	0.2070	0.1017
Cl.8	0	0	0	0	0	0	0	0	0.2498	0.6873
Cl.9	0	0	0	0	0	0	0	0	0	0.3026
Cl.10	0	0	0	0	0	0	0	0	0	0

Taux d'erreur par problème bi-classe en utilisant l'ancienne formulation

	Cl.1	Cl.2	Cl.3	Cl.4	Cl.5	Cl.6	Cl.7	Cl.8	Cl.9	Cl.10
Cl.1	0	0.0473	0.4970	0.0503	0.0510	0.4274	0.4128	0.0996	0.4606	0.3017
Cl.2	0	0	0.2307	0.0466	0.0945	0.0493	0.1433	0.2312	0.0474	0.1399
Cl.3	0	0	0	0.2449	0.3476	0.1040	0.4020	0.8252	0.3988	0.3920
Cl.4	0	0	0	0	0.1004	0.5783	0.0508	0.3925	0.4032	0.5944
Cl.5	0	0	0	0	0	0.0534	0.4124	0.2488	0.2556	0.7534
Cl.6	0	0	0	0	0	0	0.9189	0.2083	0.3215	0.4734
Cl.7	0	0	0	0	0	0	0	0.1007	0.2070	0.1017
Cl.8	0	0	0	0	0	0	0	0	0.2997	0.6873
Cl.9	0	0	0	0	0	0	0	0	0	0.3026
Cl.10	0	0	0	0	0	0	0	0	0	0

BIBLIOGRAPHIE

[1] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine Learning*, 46(1-3):131--159, 2002.

[2] G. Rätsch, T. Onoda, and K.-R. Müller, "Soft Margins for AdaBoost," *Machine Learning*, vol. 42, pp. 287 - 320, 2001.

[3] M. M. Adankon, M. Cheriet, and N. E. Ayat, "Optimizing Resources in Model Selection for Support Vector Machines," presented at International Joint Conference in Neural Networks, Montreal (to appear in July 2005), 2005.

[4] K.-M. Chung, W.-C. Kao, and L.-L. W. C.-L. Sun, and C.-J. Lin, "Radius Margin Bounds for Support Vector Machines with the RBF Kernel," *Neural Computation*, vol. 15, pp. 2643-2681, 2003.

[5] N. E. Ayat, M. Cheriet, and C. Y. Suen, "Empirical error based optimization of SVM kernels: application to digit image recognition," *International Workshop on Handwriting Recognition*, pp. 292-297, 2002.

[6] N. E. Ayat, "Sélection automatique de modèle des machines à vecteurs de support: Application à la reconnaissance d'images de chiffres manuscrits." Thèse de PhD, École de Technologie Supérieure, Montréal, 2003.

[7] G. Baudat and F. Anouar, "Kernel-based methods and function approximation," *International Joint Conference on Neural Networks*, pp. 1244-1249, 2001.

[8] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*: Cambridge University Press, 2000.

[9] J. Shawe-Taylor and N. Cristianini, "Kernel Methods for Pattern Analysis," Cambridge University Press, 2004.

[10] J. Mercer, "Functions of positive and negative type and their connection with the theory of integral equations," *Philos. Trans. Roy. Soc. London*, vol. A209, pp. 415-446, 1909

[11] N. E. Ayat, M. Cheriet, and C. Y. Suen, "Kmod-a two parameter svm kernel for pattern recognition," *International Conference on Pattern Recognition*, vol. 3, pp. 331- 334, 2002.

[12] B. Scholkopf, A. J. Smola, and K.-R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1199-1319, 1998.

[13] B. Scholkopf, A. J. Smola, and K.-R. Muller, "kernel Principal Component Analysis," in *Advances in Kernel Methods: Support Vector Machines*, Scholkopf, Burges, and Smola, Eds.: MIT Press, Cambridge, MA, 1998, pp. 327-352.

[14] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller, "Fisher discriminant analysis with kernel," in *Neural Networks for Signal Processing IX*, Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, Eds.: Piscataway, NJ:IEEE, 1999, pp. 41-48.

[15] V. Roth and V. Steinhage, "Nonlinear discriminant analysis using kernel functions," in *Advances in Neural Information Processing Systems 12*, S. A. Solla, T. K. Leen, and K.-R. Müller, Eds.: Cambridge, MA: MIT Press, 2000, pp. 568-574.

[16] G. Baudat and F. Anouar, " Generalized discriminant analysis using a kernel approach," *Neural Computation*, vol. 12, pp. 2385-2404, 2000.

[17] F. Abdallah, C. Richard, and R. Lengelle, "A Sequential approach for multi-class discriminant analysis with kernels," *ICASSP*, pp. 453-456, 2004.

[18] G. Baudat and F. Anouar., "Feature vector selection and projection using kernels," *Neurocomputing*, vol. 55, pp. 31-38, 2003.

[19] M. E. Tipping, "The Relevance Vector Machine," *Advances in Neural Information Processing Systems*, pp. 652-658, 2000.

[20] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *Machine Learning Research*, vol. 1, pp. 211-244, 2001.

[21] V. N. Vapnik, "Principles of risk Minimization for learning theory," *Adavances in Neural Information Processing Systems 4, Morgan Kaufman, San Mateo, CA*, pp. 831-838, 1992.

[22] V. N. Vapnik, *Estimation of Dependences based on Empirical Data*. Berlin: Springer Verlag, 1982.

[23] V. N. Vapnik and A. J. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," *Doklady Akademii Nauk USSR*, 1968.

[24] V. N. Vapnik and A. J. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," *Theory Probab. Apl.*, pp. 264-280, 1971.

[25] V. N. Vapnik, *Statistical learning theory*. New York: John Wiley and Sons, 1998.

[26] E. Osuna, R. Freund, and F. Girosi, "Improved training algorithm for support vector machines," presented at NNSP'97, 1997.

[27] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in Kernel Methods: Support Vector Machines*, Scholkopf, Burges, and Smola, Eds.: MIT Press, Cambridge, MA, 1998.

[28] G. Zoutendjik, "Methods of feasible directions: a study in linear and non-linear programming", Elsevier, 1970.

[29] J. Platts, " Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods: Support Vector Machines*, Scholkopf, Burges, and Smola, Eds.: MIT Press, Cambridge, MA, 1998.

[30] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to Platt's SMO algorithm for SVM classifier design," *Neural Computation*, vol. 13, pp. 637-649, 2001.

[31] T. Hastie and R. Tibshirani, "Classification by Pairwise Coupling," in *Advances in Neural Information Processing Systems*, vol. 10, Michael I. Jordan , M. J. Kearns, and a. S. A. Solla, Eds., 1998.

[32] T.-F. Wu, C.-J. Lin, and R. C. Weng, "Probability Estimates for Multi-class Classification by Pairwise Coupling," *Journal of Machine Learning Research*, vol. 5, pp. 975-1005, 2004.

[33] M. Moreira and E. Mayoraz, "Improved Pairwise Coupling Classification with Correcting Classifiers," In *Proceedings of the 10th European Conference on Machine Learning*, pages 160–171, 1998.

[34] C. Hsu and C. Lin, "A comparison of methods for multi-class support vector machines," IEEE Trans. on Neural Networks, 13(2):415--425, March 2002.

[35] G. Wahba, Y. Lin, and H. Zhang, " Generalized Approximate Cross Validation for Support Vector Machines, or, Another Way to Look at Margin-Like Quantities," Departement of Statistics,University of Wisconsin, Madison, Rapport Technique n° 1006, February 25 1999.

[36] T. S. Jaakkola and D. Haussler, "Probabilistic kernel regression models," In D. Heckerman and J. Whittaker, editors, Workshop on Arti cial Intelligence and Statistics 7. Morgan Kaufmann, 1999.

[37] M. Opper and O. Winther, "Mean field methods for classification with Gaussian processes," in Advances in Neural Information Processing Systems 11 (NIPS'98), M. S. Kearns, S. A. Solla, and D. A. Cohn, eds., MIT Press, Cambridge, MA 1999.

[38] M. Opper and O. Winther, " Gaussian processes and svm: Mean field and leave-one-out," in *Advances in Large Margin Classifiers*, A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, Eds. Cambridge: MIT Press, 2000, pp. 311-326.

[39] T. Joachims, "Estimating the generalization performance of a svm efficiently," *International Conference on Machine Learning*, pp. 431-438, 2000.

[40] O. Chapelle and V. Vapnik, "Model selection for support vector machines," *Advances in Neural Information Processing Systems*, 1999.

[41] V. Vapnick and O. Chapelle, "Bounds on error expectation for support vector machines," *Neural Computation*, vol. 12(9), 2000.

[42] K. Duan, S. Keerthi, and A. N. Poo, " Evaluation of simple performance measures for tuning SVM hyperparameters," *Neurocomputing*, vol. 51, pp. 41-59, 2003.

[43] J. Platt, "Probabilistic outputs for support vector machines and comparison to regularized likelihood methods," in *Advances in Large Margin Classiers*, A. J. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, Eds., 2000, pp. 61-74.

[44] H.-T. Lin, C.-J. Lin, and R. C. Weng., "A Note on Platt's Probabilistic Outputs for Support Vector Machines," Technical Report, National Taiwan University, May 2003.

[45] C. M. Bishop, *Neural Networks for Pattern Recognition*: Oxford University Press, Oxford, Great Britain, 1995.

[46] C. Domeniconi and D. G., "Incremental Support Vector Machine Construction," presented at International Conference on Data Mining, pp.589-592, 2001.

[47] N. A. Syed, H. Liu, and K. K. Sung, "Incremental Learning with Support Vector Machines," presented at International Joint Conference on Artificial Intelligence, 1999.

[48] P. Mitra, C. A. Murthy, and S. K. Pal, "Data Condensation in Large Databases by Incremental Learning with Support Vector Machines," *International Conference on Pattern Recognition*, pp. 712-715, 2000.

[49] M. M. Adankon, "Étude comparative entre le MLP et le SVM sur la base MNIST," Rapport technique, École de Technologie Supérieure, Montréal 2004.

[50] J.-X. Dong, A. Krzyzak, and C. Y. Suen, "Fast SVM Training Algorithm with Decomposition on Very Large Data Sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 603-618, 2005.

[51] L. Breiman, "Bias, variance, and arcing classifiers," Technical Report 460, Statistics Department, University of California, Statistics Department, University of California, Berkeley, April 1996.

[52] Y. LeCun, L. Bottom, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278-2324, 1998