ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THÈSE PRÉSENTÉE À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DU
DOCTORAT EN GÉNIE
Ph.D.

PAR
Mathias Mahouzonsou ADANKON

APPRENTISSAGE SEMI-SUPERVISÉ POUR LES SVMS ET LEURS VARIANTES

MONTRÉAL, LE 5 NOVEMBRE 2009

"Que le sage écoute, et il augmentera son savoir,

Et celui qui est intelligent acquerra de l'habileté, "

Les Proverbes.

CETTE THÈSE A ÉTÉ ÉVALUÉE

PAR UN JURY COMPOSÉ DE

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

Dr. Alain Biem, co-directeur de thèse
IBM at Waston Research Center, NY, USA

Pr. Pierre Dumouchel, président du jury
Département de génie logiciel et des technologies de l'information

Dr. Isabelle Guyon, membre externe
ClopiNet, Berkeley, CA , USA

Pr. Tony Wong, membre du jury
Département de génie de la production automatisée á l'École de technologie supérieure

ELLE A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 21 SEPTEMBRE 2009

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

En premier lieu, je tiens à remercier le Professeur Mohamed Cheriet pour m'avoir fait confiance tout au long de ce doctorat. Son suivi, son soutien et ses vifs encouragements m'ont permis de réaliser ce doctorat dans de bonnes conditions. Je remercie aussi vivement Dr Alain Biem pour tout l'encadrement dont j'ai bénéficié de sa part. Ses conseils avisés tout au long de cette recherche ont permis d'enrichir cette thèse.

De même, je remercie Dr Isabelle Guyon, Pr Tony Wong et Pr Pierre Dumouchel pour avoir pris le soin d'évaluer ce travail.

J'adresse aussi ma gratitude au Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG) pour leur soutien financier.

Un grand merci va également à l'ensemble des compagnons du LIVIA et de Synchromedia avec qui j'ai partagé bien plus qu'un lieu de travail.

Enfin, je témoigne toute ma reconnaissance à Pascaline, Godfavor et toute ma famille, Gabriel, Emmanuel, Roger, Simon, Marie et Alice, pour leur soutien inexprimable, et ma gratitude envers le Rév. Fofy Ndélo.

APPRENTISSAGE SEMI-SUPERVISÉ POUR LES SVMS ET LEURS VARIANTES

Mathias Mahouzonsou ADANKON

RÉSUMÉ

La reconnaissance de formes est un domaine fort intéressant de l'intelligence artificielle. Pour résoudre les problèmes de reconnaissance de formes, des classifieurs sont construits en utilisant des prototypes de données à reconnaître ainsi que leur classe d'appartenance. On parle d'apprentissage supervisé. Aujourd'hui, face aux importants volumes de données disponibles, le coût de l'étiquetage des données devient très exorbitant. Ainsi, il est impraticable, voir impossible d'étiqueter toutes les données disponibles. Mais puisque, nous savons que la performance d'un classifieur est liée au nombre de données d'apprentissage, la principale question qui ressort est comment améliorer l'apprentissage d'un classifieur en ajoutant des données non étiquetées à l'ensemble d'apprentissage. La technique d'apprentissage issue de la réponse à cette question est appelée apprentissage semi-supervisé.

La machine à vecteurs de support(SVM) et sa variante Least-Squares SVM (LS-SVM) sont des classifieurs particuliers basés sur le principe de la maximisation de la marge qui leur confère un fort pouvoir de généralisation. Au cours de nos travaux de recherche, nous avons considéré l'apprentissage semi-supervisé de ces machines. Dès lors, nous avons proposé diverses techniques d'apprentissage de ces machines pour accomplir cette tâche.

Dans un premier temps, nous avons utilisé l'inférence bayésienne pour estimer les paramètres du modèle et les étiquettes. Ainsi, nous avons élaboré des formulations bayésiennes à un et deux niveau(x) d'inférence, qui sont par la suite appliquées aux SVMs et aux LS-SVMs dans le contexte de l'apprentissage semi-supervisé.

Dans un second temps, nous avons proposé d'améliorer la technique d'auto-apprentissage, en utilisant un classifieur d'approche générative pour aider le principal classifieur discriminant entraîné en semi-supervisé à étiqueter les données. Nous nommons cette stratégie *Apprentissage soutenu (Help-Training)*, et nous l'avons appliqué avec succès aux SVMs et à sa variante LS-SVM.

Nos divers algorithmes d'apprentissage semi-supervisé ont été testés sur des données artificielles et réelles et ont donné des résultats encourageants. Cette validation a été appuyée par une analyse montrant les avantages et les limites de chacun des méthodes développées.

Mots-clés : apprentissage semi-supervisé, machines à vecteurs de support, SVM, LS-SVM.

SEMI-SUPERVISED LEARNING FOR SVMS AND THEIR VARIANTS

Mathias Mahouzonsou ADANKON

ABSTRACT

Pattern recognition problems are solved using classifiers which are machines built by using prototypes of the data to be recognized. Traditionally, machine learning needs labeled data, as does the training procedure called supervised learning. Now, labeling, as well as the collection of the data itself, is a process which requires considerable human effort and is therefore very expensive. However, since we know that the larger the number of training samples, the better the performance of the classifier -making assumption that we have a correct classifier model-, the issue becomes finding a way to improve supervised learning by adding unlabeled data. Training using both labeled and unlabeled data is called semi-supervised learning . In this type of training, the classifier is built by learning from both labeled and unlabeled data. Hence, it is not necessary to label all the data collected in order to train the classifier.

Support vector machine (SVM) and its variant Least-Squares SVM (LS-SVM) are particular discriminative classifiers based on the margin-maximization performing structural risk and have excellent power of generalization. In this work, we consider its use in semi-supervised learning and propose various algorithms to perform training task for these machines.

First, we use Bayesian approach to infer both the model parameters and the labels for unlabeled data. We develop a one- and two-level Bayesian inference schemes, and perform their application for learning semi-supervised SVM and semi-supervised LS-SVM.

Second, we propose to reinforce self-training strategy by using a generative classifier that may help the main discriminative classifier training in semi-supervised mode to label the unlabeled data. We called this semi-supervised strategy : Help-training, which is applied for training semi-supervised SVM and LS-SVM.

Experimental results on both artificial and real problems show the usefulness of our different semi-supervised algorithms. Depth analysis of the experimental results reveals the advantages and the limits of each method.

Keywords : semi-supervised learning, support vector machine, SVM, LS-SVM.

TABLE DES MATIÈRES

Page

LISTE DES FIGURES

Page

LISTE DES TABLEAUX

Page

LISTE DES ALGORITHMES

LISTE DES ABBRÉVIATIONS

ACP	**A**nalyse en **C**omposantes **P**rincipales
AG	**A**lgorithme **G**énétique
$BS^2LS-SVM$	**B**ayesian **S**emi-**S**upervised **L**east **S**quares **S**upport **V**ector **M**achines
EM	**E**xpectation **M**aximization
FSALS-SVM	**F**ast **S**parse **A**pproximation for LS-SVM
HTS^3VM	**H**elp **T**raining **S**emi-**S**upervised **S**upport **V**ector **M**achines
$HTS^2LS-SVM$	**H**elp **T**raining **S**emi-**S**upervised **L**east **S**quares **S**upport **V**ector **M**achines
IA	**I**ntelligence **A**rtificielle
KKT	**K**arush **K**uhn **T**ucker
KPCA	**K**ernel **P**rincipal **C**omponent **A**nalysis
LS-SVM	**L**east **S**quares **S**upport **V**ector **M**achines
LOO	**L**eave-**O**ne-**O**ut
MAP	**M**aximum **A** **P**osteriori
QP	**Q**uadratic **P**rogramming
SVM	**S**upport **V**ector **M**achines
S^3VM	**S**emi-**S**upervised **S**upport **V**ector **M**achines
S^3VM-GA	**S**emi-**S**upervised **S**upport **V**ector **M**achines **G**enetic **A**lgorithm
$S^2LS-SVM$	**S**emi-**S**upervised **L**east **S**quares **S**upport **V**ector **M**achines
TSVM	**T**ransductive **S**upport **V**ector **M**achines
USPS	**U**nited **S**tates **P**ostal **S**ervice

LISTE DES NOTATIONS

$p(x, y)$	probabilité jointe de x et y
$p(x\|y)$	probabilité conditionnelle de x sachant y
x_i	vecteur caractéristique de la i-ème observation
y_i	étiquette associée à la i-ème observation
$\mathbb{R}$	l'ensemble des nombres réels
w'	transposée du vecteur w
θ	paramètre du classifieur
λ	hyperparamètre du classifieur
ϕ	fonction de projection $\phi : \mathbb{R}^d \rightarrow F$
F	espace de grande dimension (feature space)
k	fonction noyau
K	Matrice noyau
log	fonction logarithme de base e avec $log(e) = 1$.
log_{10}	fonction logarithme de base 10 avec $log_{10}(10) = 1$.
$\vec{1}$	vecteur rempli de 1

INTRODUCTION

L'intelligence artificielle est sans doute l'un des domaines de recherches scientifiques qui a connu une forte avancée ces dernières années. Souvent abrégée par le sigle IA, l'intelligence artificielle est définie par l'un de ses créateurs, Marvin Lee Minsky, comme « la construction de programmes informatiques qui s'adonnent à des tâches qui sont, pour l'instant, accomplies de façon plus satisfaisante par des êtres humains car elles demandent des processus mentaux de haut niveau tels que : l'apprentissage perceptuel, l'organisation de la mémoire et le raisonnement critique » [1]. L'intelligence artificielle utilise des ordinateurs (avec des programmes informatiques adéquats) ou de processus électroniques élaborés afin d'imiter le comportement humain. Donc, il s'agit de reproduire l'intelligence humaine mais avec l'utilisation des machines, d'où la qualification artificielle. On parle aussi de doter des machines de capacités intellectuelles comparables à celles des êtres humains [2]. Les domaines traités par l'IA en général couvrent les sciences cognitives, la représentation et l'acquisition des connaissances, la robotique, la vision par ordinateur, la reconnaissance de formes, l'apprentissage, le langage naturel, la modélisation des raisonnements, ...

La reconnaissance de formes (écriture, texte, parole, visages, empreintes digitales, diagnostics cliniques, etc) qui est l'un des axes de recherche de l'IA, s'occupe du problème d'assignation d'une donnée à une classe (forme). Cette discipline désigne l'ensemble des techniques et méthodes développées dans le but d'identifier des objets et de les assigner à des classes : on parle aussi de la classification. Cette branche de l'IA s'occupe donc de la conception et du développement des machines informatiques capables d'identifier des formes. C'est un domaine fort intéressant et il trouve son application dans de nombreux systèmes d'interfaces visuelles, d'analyse de données (biomédicales), de bioinformatique, de multimédia, etc. Les recherches dans ce domaine se subdivisent en deux secteurs connexes et dépendants quant au résultat fi-

1. *http : //encyclopedie.snyke.com/articles/intelligence_artificielle.htm*
2. Cependant, malgré le pouvoir de calcul très puissant des machines modernes, l'homme ne peut-être comparé à une machine

nal. Il s'agit de l'extraction de caractéristiques et de la construction des classifieurs.

L'extraction de caractéristiques regroupe les méthodes développées pour décrire mathématiquement les formes (les objets) que l'on désire classifier. C'est une étape de pré-traitement pour la reconnaissance de formes. Elle est souvent décomposée en processus de construction et de sélection de caractéristiques. Son but ultime est de fournir des descriptions mathématiques suffisamment riches des objets à reconnaître pour mieux faciliter le travail des classifieurs qui se retrouvent en amont. Les algorithmes conçus pour réaliser ce but prennent souvent en entrée des données unidirectionnelles (signal audio) ou multidimensionnelles (images 2D, 3D) issues des capteurs utilisés pour capturer la forme des objets. Cependant, la construction des classifieurs est dévouée aux techniques de conception des fonctions de décision pour attribuer les classes aux objets, une fois transformés sous forme de modèles mathématiques.

Problématique de la recherche

Pour résoudre les problèmes de reconnaissance de formes, des classifieurs (des machines) sont construits en utilisant des prototypes des données à reconnaître. Traditionnellement, l'apprentissage automatique a besoin des données étiquetées qui nécessitent beaucoup d'effort humain pour réaliser non seulement la collecte mais aussi l'étiquetage. Ainsi, les données collectées non étiquetées ne sont pas utiles dans ce mode d'apprentissage qualifié de *supervisé*. En apprentissage automatique, l'apprentissage supervisé désigne l'ensemble des techniques utilisées pour produire une fonction de prédiction permettant d'associer un objet à un nombre entier (classification) ou à un nombre réel (régression), en se basant sur une base d'apprentissage contenant les données et leurs étiquettes. Le deuxième type d'apprentissage est l'apprentissage *non-supervisé* où l'on cherche à trouver l'organisation des données en se basant sur des données non-étiquetées. Ce type d'apprentissage ressemble au problème d'estimation de densité en statistiques et ne permet pas pour autant de mettre en œuvre des classifieurs. Ainsi, l'idéal pour construire des classifieurs performants dédiés à une tâche spécifique et prédéfinie

est la conception des algorithmes d'apprentissage supervisé.

Mais le processus de l'étiquetage des données devient très coûteux ou très difficile à réaliser (par exemple l'étiquetage de la parole, des images ou des pages web nécessite une perspicacité humaine tandis que dans le domaine de la médecine ou de la biologie, on a besoin quelques fois de plusieurs tests ou expériences très complexes), avec l'augmentation des données disponibles pour certaines applications. Alors, l'étiquetage de toutes les données disponibles s'avère très dispendieux, voir impossible. Pour y remédier, on a introduit la notion d'apprentissage semi-supervisé qui consiste à modéliser le classifieur en se basant sur des données étiquetées et sur celles non étiquetées. Cette troisième technique d'apprentissage, malgré qu'elle permette de minimiser le coût d'étiquetage sans beaucoup nuire à la performance du classifieur, est peu abordée dans la recherche par rapport aux autres formes d'apprentissage.

Parmi les classifieurs, se distinguent ceux qui utilisent la méthode des noyaux (kernel method) qui est une technique récente en reconnaissance de formes et plus généralement en algorithmes d'apprentissage (machine learning). Elle consiste à projeter les données qui sont initialement non linéairement 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 ou séparateurs à vaste marge (SVMs), issues des travaux de Vapnik, utilisent cette technique de noyau 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 [83]. 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.

Les SVMs ont suscité l'intérêt de plusieurs communautés de recherche. Il y a eu des travaux de recherche sur les algorithmes d'apprentissage en mode supervisé [61; 45; 63], la sélection de modèle [43; 60; 59; 23; 24; 79; 7; 3], etc. Aussi, plusieurs applications ont été développées avec succès avec les SVMs en classification comme en régression. On peut citer notamment la reconnaissance de l'écriture manuscrite, la classification de texte, la reconnaissance d'objets 3D, le diagnostic de maladies, la classification de gènes, de molécules, la planification financière, etc [3].

Le "least-squares support vector machine" (LS-SVM) est une variante du standard SVM. Il découle de la question suivante :"Comment peut-on simplifier la formulation des SVMs sans perdre aucun de leurs avantages ?" En répondant à cette question, Suykens et Vandewalle [72] ont proposé les LS-SVMs où l'apprentissage consiste à résoudre un problème convexe comme dans le cas des SVMs. Aussi, il a été démontré par une étude expérimentale minutieuse que les LS-SVMs et les SVMs ont des performances de généralisation comparable [77]. Ainsi, les LS-SVMs peuvent être considérés comme d'excellents remplaçants des SVMs. De plus, l'algorithme d'apprentissage des LS-SVMs consiste à résoudre un système linéaire à l'instar des SVMs où l'on résout un problème de programmation quadratique.

Malgré que les SVMs et les LS-SVMs ont eu beaucoup de succès en classification, les recherches pour modéliser ces classifieurs en mode semi-supervisé sont limitées. Au début de notre thèse, aucun travail n'était réellement fait sur l'apprentissage semi-supervisé de ces machines, excepté le travail de Vapnik concernant la transductive SVM qui fut implémenté par Joachims [48]. Depuis, les quelques algorithmes proposés souffrent des problèmes de performance, de garantie de convergence (fonction objective non-convexe), de "scalability" (problème survenu lorsque la taille des données devient importante)[21], etc. Ainsi, le développement d'algorithmes d'apprentissage semi-supervisé pour ces machines s'avère un travail de recherche intéressant.

3. Pour plus de détails et surtout les références de ces applications, voir le site http ://www.clopinet.com/isabelle/Projects/SVM/applist.html

Objectifs de la recherche

Durant nos travaux de recherche, nous avons proposé de développer des algorithmes d'apprentissage pour les SVMs et ses variantes comme les LS-SVMs en mode semi-supervisé. Nous avons développé des techniques algorithmiques efficaces afin d'utiliser cette forme d'apprentissage avec les SVMs et ses variantes de façon promettante dans des applications réelles. Ainsi, les stratégies qui sont développées au cours de nos travaux de recherche permettront de réaliser l'apprentissage des SVMs/LS-SVMs avec des données étiquetées et non-étiquetées, ce qui permettra d'étendre l'utilisation des machines à vecteurs de support et ses variantes à d'autres applications, soit en mode statique ou en mode dynamique. Dans ce dernier mode, le classifieur résultant a une autonomie d'apprentissage et d'adaptation face aux nouvelles données qui arrivent dans le système.

Pour réaliser l'apprentissage des SVMs et de ses variantes en mode semi-supervisé, nous nous sommes fixés les objectifs spécifiques suivants :

(i) Élaborer une ou plusieurs formulation(s) pour l'apprentissage semi-supervisé dans un cadre de travail général ;

(ii) Appliquer le ou les modèle(s) développé(s) en *(i)* aux machines à vecteurs de support et à ses variantes ;

(iii) Valider les approches ainsi développées sur des applications réelles (classification de texte et reconnaissance d'écriture manuscrite).

Organisation de la thèse

Cette thèse est articulée autour de trois points. Le premier est la formulation bayésienne pour l'apprentissage semi-supervisé. Le second concerne le développement de la stratégie qualifiée

de "Help-Training" pour le même type d'apprentissage. Enfin, le dernier point se rapporte à la sélection de modèle afin d'améliorer la performance des classifieurs ainsi développés.

Le présent manuscrit est organisé en cinq chapitres, comme suit :

Le premier chapitre est consacré à l'état de l'art. La partie bibliographique de cette thèse est organisée en deux parties. Dans la première partie, nous rapportons les différentes méthodes de l'apprentissage semi-supervisé. Des techniques développées de façon générale pour tous les types de classifieurs mais qui sont adaptées au mieux à une classe spécifique de classifieurs. Parmi ces méthodes, nous citons : le modèle de reproduction, l'auto-apprentissage, le co-apprentissage. Dans la deuxième partie, nous nous intéresserons aux classifieurs auxquels seront appliquées les différentes méthodes d'apprentissage semi-supervisé que nous avons développées. Ce sont les SVMs et leurs variantes les LS-SVMs.

Dans le second chapitre, nous présentons la formulation bayésienne que nous avons proposé pour résoudre le problème d'apprentissage semi-supervisé. Il s'agit d'utiliser le processus d'inférence bayésienne afin d'estimer les paramètres du modèle et les classes d'appartenance des données non étiquetées. Sachant que l'inférence bayésienne est fondée sur la manipulation d'énoncés probabilistes avec la formule de Bayes, nous avons développé un cadre de travail général et concis. De façon spécifique, nous proposons dans cette formulation l'inférence à un niveau et celle à deux niveaux. De plus, dans ce chapitre, nous avons présenté l'application de la formulation bayésienne avec les SVMs et les LS-SVMs. D'une part nous avons montré comment retrouver des algorithmes d'apprentissage semi-supervisé existants en utilisant notre formulation bayésienne et d'autre part, nous avons développé de nouveaux algorithmes d'apprentissage pour les SVMs et les LS-SVMs en mode semi-supervisé.

Dans le troisième chapitre de cette thèse nous nous intéressons à la stratégie d'apprentissage semi-supervisé développée spécifiquement pour les classifieurs utilisant l'approche discriminante. Cette technique utilise un classifieur de type génératif pour *aider* le classifieur principal (discriminant) à faire un bon auto-étiquetage. Nous donnons l'appellation de *Help-Training* à cette méthode. Nous présentons également dans cette partie de la thèse, l'application de cette stratégie d'apprentissage semi-supervisé aux SVMs et aux LS-SVMs.

Le quatrième chapitre présente la sélection de modèle pour les SVMs/LS-SVMs. Plusieurs travaux sur la sélection de modèle des SVMs sont disponibles [84; 83; 43; 46; 60; 59; 23; 79; 36], de plus nos travaux de recherche antérieurs ont traité de ce sujet [5; 2; 3]. Ainsi, pour cette thèse, nous avons consacré nos efforts pour améliorer la sélection automatique des hyperparamètres des LS-SVMs. De plus, nous avons proposé une stratégie pour réaliser la sélection de modèle en mode semi-supervisé pour les SVMs/LS-SVMs.

Le dernier chapitre de cette thèse est consacré aux expérimentations et à la validation de nos résultats. Puisque, nous avons proposé des solutions au problème d'apprentissage semi-supervisé en développant plusieurs approches, les différents algorithmes conçus seront validés en utilisant le même protocole expérimental afin de faciliter une comparaison saine et concise. De plus, nous avons comparé nos algorithmes avec d'autres algorithmes destinés pour les mêmes tâches.

Contributions

Le chapitre 2 de cette thèse a fait l'objet de trois articles.

(i)"Genetic Algorithm based training for Semi-supervised SVM". Soumis à Neural Computing and Applications.

(ii)"Semi-Supervised Least Squares Support Vector Machine". Sous presse à IEEE Transactions on Neural Networks.

(iii)"Semi-Supervised Learning using Bayesian inference". Soumis à IEEE Transactions on Pattern Analysis and Machine Intelligence.

Les travaux du chapitre 3 ont été présentés partiellement à *International Conference on Pattern Recognition 2008* et à *International Joint Conference on Neural Networks 2008* ; puis la version enrichie est soumise à Pattern Recognition Journal.

(iv) "Help-Training for Semi-supervised Learning". Soumis à Pattern Recognition Journal.

Enfin, le chapitre 4 a fait l'objet d'une publication dans Pattern Recognition Journal.

(v) "Model Selection for LS-SVM : Application to Handwriting Recognition". Pattern Recognition 42(12) : 3264-3270 (2009).

CHAPITRE 1

ÉTAT DE L'ART

1.1 Introduction

Le présent chapitre est consacré à la revue de littérature générale sur le thème abordé dans cette thèse et est organisé en deux principales parties. Une première partie qui rapporte les diverses techniques d'apprentissages semi-supervisé proposées dans la littérature. Nous allons plus mettre l'emphase sur les méthodes développées d'une façon générale et qui ne sont pas spécifiques à un classifieur déterminé. Puis, la deuxième partie de cette revue de littérature est dédiée aux SVMs et ses variantes qui sont des classifieurs robustes avec un fort pouvoir de généralisation.

1.2 Aperçu sur l'apprentissage semi-supervisé

Plusieurs sortes de techniques ont été développées pour réaliser la tâche de l'apprentissage semi-supervisé. Dans cette section, nous allons décrire en bref certaines catégories de ces techniques qui sont présentées dans [92]. Avant de présenter ces techniques, nous allons rappeler certaines hypothèses qui permettent le bon fonctionnement des algorithmes d'apprentissage semi-supervisé [20]. Considérons les données d'apprentissage x_i et l'étiquette associé y_i.

A. Hypothèse de continuité

Elle s'énonce comme suit : si deux points x_1 et x_2 sont proches dans une région de forte densité, alors il en est de même pour leurs étiquettes y_1 et y_2. Cette hypothèse permet de préserver la continuité de la fonction de classification ou de régression.

B. Hypothèse de regroupement

Elle est une conséquence de la précédente pour le cas de la classification : si deux points x_1 et x_2 sont dans un même regroupement (cluster), alors ils sont de la même classe $y_1 = y_2$. Mais, cela ne signifie pas que chaque classe doit être représentée par un regroupement.

C. Hypothèse de "Manifold"

"Manifold" désigne l'espace mathématique construit sur une mesure de similarité c'est-à-dire chaque donnée n'a que des données qui lui ressemblent dans son voisinage. Cette technique permet de contrôler la notion de proximité surtout lorsque l'espace des caractéristiques est trop grand et de combattre la malédiction de la dimensionnalité. Cette hypothèse est au cœur des méthodes basées sur les graphes : chaque classe se retrouve sur un "Manifold" séparé.

1.2.1 Modèle de reproduction (*Generative Model*)

C'est la plus ancienne des techniques pour l'apprentissage semi-supervisé. L'idée de base consiste à considérer un model $p(x, y) = p(x|y)p(y)$ où $p(x|y)$ est un mélange de distributions, par exemple un mélange de gaussiennes. Ainsi, avec les données étiquetées et non étiquetées, on caractérise les différentes composantes du modèle de distribution supposé. Puis avec un seul élément étiqueté de chaque composante, on complète les caractéristiques du mélange. Dans [29; 49; 16; 64], on retrouve cette technique utilisée sous plusieurs formes et souvent avec des algorithmes EM (Expectation-Maximization), qui sont conçus pour trouver le maximum de vraisemblance des paramètres de modèle probabiliste lorsque ce dernier dépend de variables latentes non observables. La figure 1.1 illustre un exemple avec des données artificielles.

Nous pouvons aussi mentionner l'utilisation de l'apprentissage non-supervisé pour faire du semi-supervisé en utilisant l'analyse de partitionnement de données (ou cluster en anglais) pour former des groupes les plus contrastés possibles. Puis, les données étiquetés sont utilisées pour identifier la classe d'appartenance de chaque groupe (voir Figure 1.2). Cette technique est utilisée avec succès dans [33] avec l'algorithme génétique.

Cependant, le véritable problème des techniques basées sur le modèle de reproduction est qu'il est très important de supposer des modèles de mélange qui reflètent la structure réelle des données, puisqu' avec une mauvaise hypothèse sur la distribution des données, les données non étiquetées affaiblissent la capacité du classifieur résultant au lieu de l'améliorer.

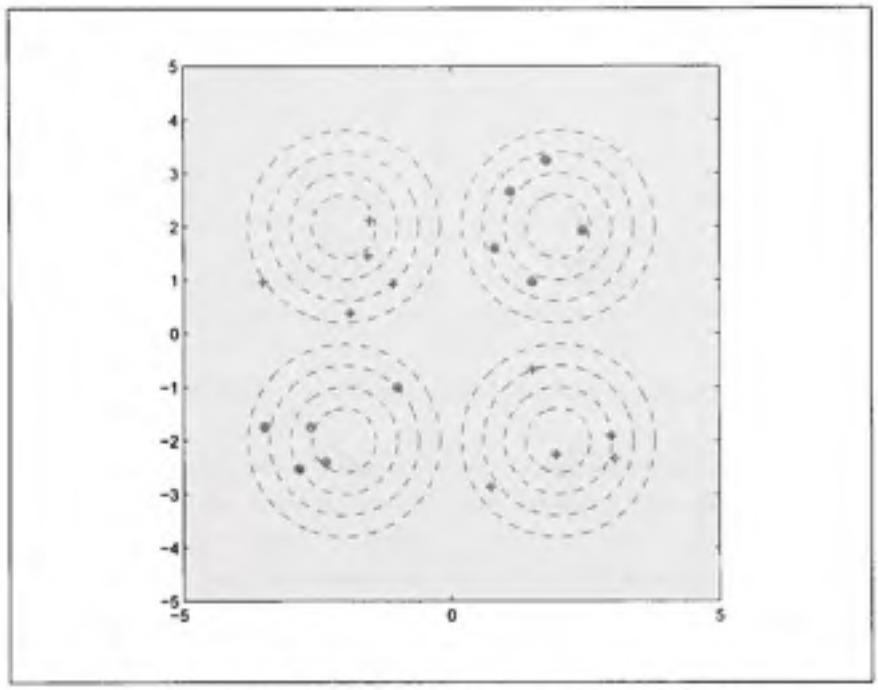


Figure 1.1 Modèle de reproduction en utilisant un algorithme EM pour identifier un mélange de 4 gaussiennes ; '+','o' et '.' représentent respectivement les données étiquetées de la classe 1 , de la classe 2 et celles qui sont non étiquetées.

1.2.2 Auto-apprentissage (*Self-Training*)

C'est une technique très répandue pour faire de l'apprentissage semi-supervisé. Le classifieur est d'abord entraîné par les données étiquetées et on utilise le résultat pour classer les données non étiquetées. Les données non étiquetées qui sont classées avec moins d'ambigüité (par exemple pour un classifieur à sortie probabiliste, on sélectionne des données qui ont une forte probabilité de classement), sont ajoutées aux données étiquetées pour former l'ensemble d'ap-

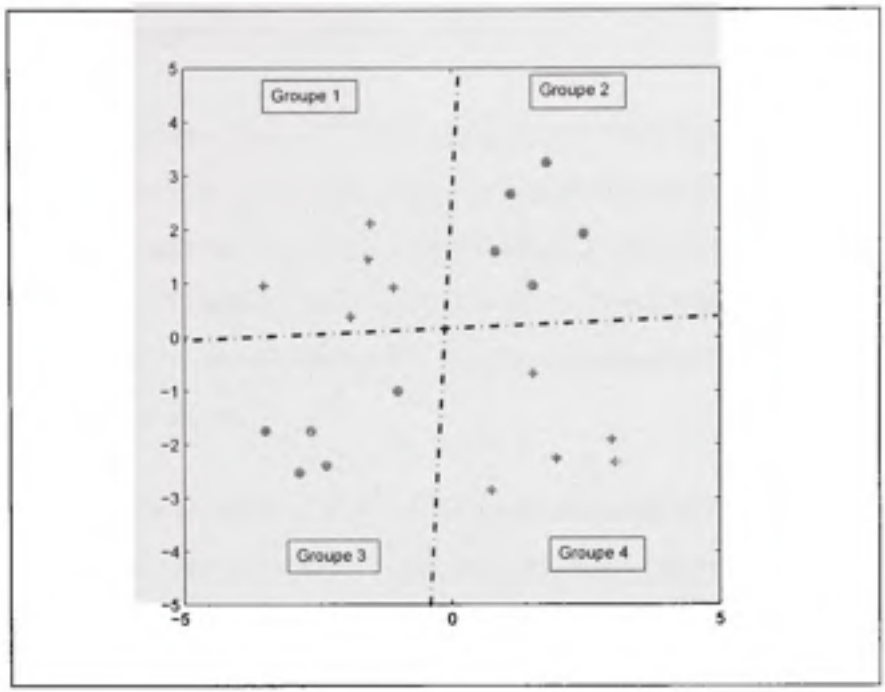


Figure 1.2 Modèle de reproduction en utilisant un algorithme de regroupement (clustering) pour identifier 4 groupes de données ; '+','o' et '.' représentent respectivement les données étiquetées de la classe 1 , de la classe 2 et celles qui sont non étiquetées.

prentissage ; puis on répète la procédure. Nous notons que cette technique utilise ses propres prédictions pour s'améliorer à chaque itération. On parle alors de "self-teaching" [66; 65]. Ce processus d'auto-étiquetage peut constituer une faiblesse pour cette technique, surtout dans le cas où la frontière de décision temporaire utilisée est très loin de la frontière réelle.

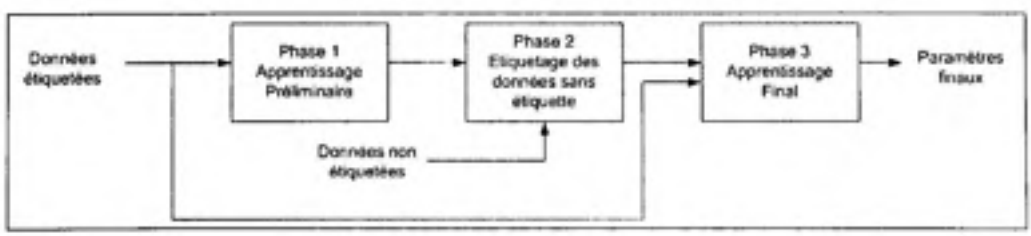


Figure 1.3 Représentation schématique de l'auto-apprentissage.

1.2.3 Co-apprentissage(*Co-Training*)

La technique de Co-apprentissage [12; 57] est basée sur l'idée selon laquelle, l'espace des caractéristiques peut être divisé en deux sous-espaces procurant chacun un bon cadre d'apprentissage. Ainsi, initialement deux classifieurs sont entraînés avec les données étiquetées sur deux sous-espaces différents. Puis chaque classifieur obtenu pour chaque sous-espace, est utilisé pour déterminer la classe probable des données non étiquetées qui seront utilisées pour re-entraîner l'autre classifieur.

Pour utiliser la méthode co-training, il faut avoir deux différentes vues [1] des données à classer, et ces deux différentes vues doivent être *compatibles* et *indépendantes*. Ces deux notions sont essentielles pour que la méthode co-training fonctionne comme c'est démontré dans [12]. La *compatibilité* permet d'avoir la même étiquette pour un exemple donné selon chaque vue considérée indépendamment. En ce qui concerne l'*indépendance*, on veut pour un exemple donné, qu'il n'y ait aucune corrélation entre les caractéristiques issues des deux différentes vues [2]. Une dernière hypothèse pour le bon fonctionnement du co-training, est que chaque vue doit être suffisamment consistante pour faire l'apprentissage d'un classifieur de façon autonome.

La méthode co-training diffère de l'auto-apprentissage ou de la technique du modèle de reproduction par le fait d'abord qu'il faut deux différentes vues sur les données avec les hypothèses nécessaires, et de plus on prend deux classifieurs pour les combiner dans la suite pour la phase de test. Ce qui constitue un avantage pour cette méthode, et l'effet supplémentaire que peut avoir cette technique est la minimisation du bruit puisqu'on dispose de deux sources indépendantes pour analyser chaque exemple. Cependant, la nécessité d'avoir deux vues sur les

1. Chaque vue désigne la façon utilisée pour extraire les caractéristiques, donc chaque vue donne lieu à une caractérisation différente des formes à reconnaître.
2. Dans le monde réel, il est difficile de réunir ces conditions. L'exemple étudié par les auteurs du co-training est la classification des pages web, vue V1 : les mots servant d'hyperlien pointant sur les pages et vue V2 : les mots se trouvant dans les pages web

données avec les contraintes citées plus haut, freine l'application de cette méthode à des problèmes réels.

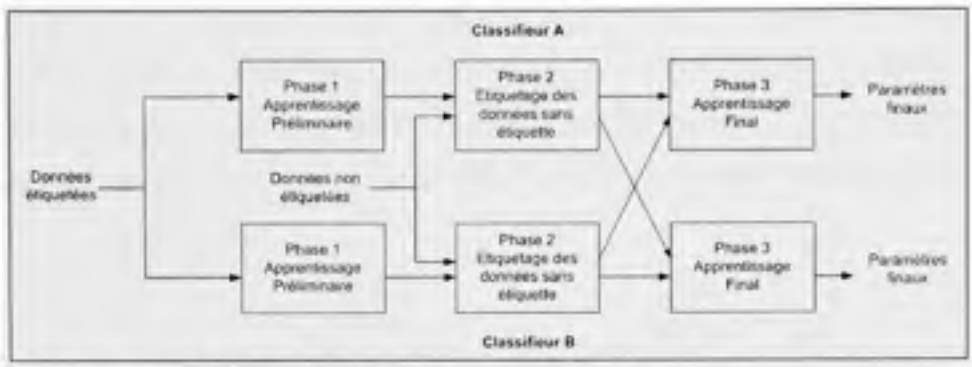


Figure 1.4 Représentation schématique du co-apprentissage.

1.2.4 Méthodes basées sur les graphes

L'apprentissage semi-supervisé basé sur les graphes consiste à définir un graphe où les sommets représentent les données d'apprentissage (étiquetées et non étiquetées). Les arêtes avec des poids w_{ij} de ce graphe reproduisent la similarité entre les données. Le processus d'étiquetage des données non étiquetées est alors réalisé en se basant sur l'hypothèse de la continuité, c'est-à-dire lorsque deux données sont proches (similarité), il en est de même pour les étiquettes correspondantes.

Les diverses méthodes de graphes existantes dans la littérature [1; 47; 93; 91; 8; 9; 25] diffèrent surtout les unes des autres par le choix de la fonction perte et du terme de la régularisation. Cependant, le succès d'un algorithme d'apprentissage basé sur les graphes dépend plus de la manière de construire ces derniers. Il y a plusieurs types d'approches :

1. Plus proches voisins. Elle consiste à connecter chaque élément du graphe aux plus k-proches-voisins en utilisant une mesure de distance pour définir les poids associés aux arêtes. La fonction Gaussienne est le plus souvent utilisée pour estimer les poids entre deux sommets [74; 93].

$$w_{ij} = exp(-\frac{\|x_i - x_j\|^2}{\sigma^2}) \tag{1.1}$$

2. Adaptation locale. Elle se base sur les informations locales pour déterminer les poids des arcs (connections) du graphes. Par exemple dans [85] , il a été supposé que chaque donnée peut être reconstruite comme la combinaison linéaire de ses voisins. Alors, les poids w_{ij} sont déterminés en minimisant l'écart d'erreur ϵ entre les données réelles et celles reconstruites.

$$\min_{w_{ij}} \left[\sum_{i=1} \left\| x_i - \sum_{j:x_j \in \mathcal{N}(x_i)} w_{ij} x_j \right\|^2 \right] \tag{1.2}$$

sous les contraintes : $\sum_i w_{ij} = 1$ et $w_{ij} \geq 0$, où $\mathcal{N}(x_i)$ représente le voisinage de x_i.

3. Injection de la connaissance du problème. Il s'agit de construire les graphes en utilisant les connaissances *a priori* du problème. Balcan et al. [55] ont fait une application de cette technique pour construire des graphes pour un problème de vidéo surveillance.

1.2.5 Autres méthodes

Plusieurs autres méthodes existent pour faire de l'apprentissage semi-supervisée. On peut citer la transduction proposée par Vapnik (Transductive SVM) [3], qui est une technique spécifique pour entraîner de façon semi-supervisée les machines à vecteurs de support. Nous avons aussi des méthodes basées sur la régularisation de l'information [47], sur les arbres [52], sur la minimisation de l'entropie [37; 75], etc.

3. Plus de détails sur Transductive SVM dans la sous-section 1.3.3

1.3 Les machines à vecteur de support et ses variantes

1.3.1 Classifieur linéaire et méthode des noyaux

Considérons un problème binaire et soit $\{(x_1, y_1); ...; (x_\ell, y_\ell)\}$ l'ensemble d'apprentissage avec $x_i \in \mathbb{R}^d$ et $y_i \in \{-1, 1\}$. Un classifieur, pour un problème binaire, est une fonction $f_\theta(x) = y : \mathbb{R}^d \to \{-1, 1\}$ qui permet d'associer à une observation x donnée une étiquette (classe). Selon la théorie de l'apprentissage, la complexité d'une fonction $f_\theta(x)$ de classification a un grand impact sur la performance de celle-ci. Car, en général, une fonction très complexe permet de modéliser parfaitement les données d'apprentissage mais en démontre une pauvreté en terme de généralisation [4] sur les données de test. Alors, souvent, on recommande de choisir des fonctions moins complexes. Dans ce cas-ci, nous considérons la famille des classifieurs linéaires :

$$f_\theta(x) = sign[w'x + b] \tag{1.3}$$

où w' représente la transposée du vecteur w, b est le biais et $\theta = (w, b)$ est le paramètre du modèle.

Cette famille de classifieurs est considérée comme des séparateurs de l'espace de caractéristiques par un hyperplan. L'espace engendré par $\mathbb{R}^d$ est séparé en deux : une partie pour les exemples dont les étiquettes sont positives et l'autre pour les exemples d'étiquettes négatives. On parle d'hyperplan séparateur.

Cependant, dans la vie réelle, les données ne sont pas linéairement séparables. Alors, les classifieurs linéaires utilisent la méthode des noyaux (kernel trick) pour définir les frontières de décisions non-linéaires.

4. Un classifieur a un fort pouvoir de généralisation lorsqu'il est capable de classer correctement des données autres que celles d'apprentissage.

La méthode des noyaux [30; 67; 68] consiste à projeter les données de l'espace $\mathbb{R}^d$ 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 $\phi : \mathbb{R}^d \rightarrow F$ est une transformation non linéaire, qui est évaluée implicitement. En réalité, on définit une fonction $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ avec $k(x_i, x_j) = \phi(x_i).\phi(x_j)$ pour réaliser la projection. La fonction k ainsi définie est appelée *noyau* et elle représente le produit scalaire dans l'espace F.

Pour définir l'existence d'une fonction noyau, on se sert souvent du théorème de Mercer [56] qui s'énonce comme suit :

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

$$\int_{\mathbb{X}} g(u)^2 du \text{ existe}$$

on a

$$\int_{\mathbb{X}\times\mathbb{X}} k(u, v)g(u)g(v)dudv \geq 0$$

La positivité de l'intégrale permet de définir l'existence du noyau, qui est un produit scalaire dans un espace de Hilbert. Le tableau 1.1 donne des exemples de noyaux.

1.3.2 Machines à vecteur de support

Les machines à vecteurs de support sont des machines linéaires particulières, basées sur le critère de la maximisation de la marge qui leur donne un excellent pouvoir de généralisation [79; 80; 81; 83]. Pour les problèmes non linéaires, les SVMs se servent de la méthode

Tableau 1.1 Exemples de noyaux

Noyau	**Expression**
Gaussien (RBF)	$k(x,y) = exp(-\|x-y\|/\sigma^2)$
Polynomial	$k(x,y) = (ax.y+b)^n$
Laplacien	$k(x,y) = exp(-a\|x-y\|+b)$
Multi-quadratique	$k(x,y) = (a\|x-y\|+b)^{1/2}$
Multi-quadratique inverse	$k(x,y) = (a\|x-y\|+b)^{-1/2}$
KMOD	$k(x,y) = a\left[exp\left(\frac{\gamma^2}{\|x-y\|^2+\sigma^2}\right)-1\right]$

des noyaux pour définir des frontières de décisions non-linéaires dans l'espace initiales des caractéristiques [14].

Une machine à vecteur de support est représentée par la fonction de décision :

$$f(x) = sign[w'\phi(x)+b] \tag{1.4}$$

dont les paramètres sont déterminés en résolvant le problème d'optimisation ci-après qui exprime la maximisation de la marge $2/\|w\|$ avec une pénalisation des erreurs de classifications.

$$\min_{w,b,\xi} \frac{1}{2}w'w + C\sum_{i=1}^{\ell}\xi_i \tag{1.5}$$

$$\text{sous les contraintes} : y_i[w'\phi(x)+b] \geq 1-\xi_i \quad \forall i = 1,...,\ell \tag{1.6}$$

$$\xi_i \geq 0 \quad \forall i = 1,...,\ell \tag{1.7}$$

Ce problème d'optimisation désigne le problème associé à l'apprentissage des SVMs de morme L1. Quant aux SVMs de norme L2, nous avons le problème ci-après :

$$\min_{w,b,\xi} \frac{1}{2} w'w + C \sum_{i=1}^{\ell} \xi_i^2 \tag{1.8}$$

$$\text{sous les contraintes } : y_i[w'\phi(x_i) + b] \geq 1 - \xi_i \quad \forall i = 1, ..., \ell \tag{1.9}$$

$$\xi_i \geq 0 \quad \forall i = 1, ..., \ell. \tag{1.10}$$

Dans la littérature, c'est le SVM de norme L1 qui est le plus courant et la plupart des implémentations d'apprentissage ne tiennent pas compte de la norme L2. Le Lagrangien du problème d'optimisation défini par (1.5) est la suivante :

$$\mathcal{L}(w, b, \xi, \alpha, \lambda) = \frac{1}{2} w'w + C \sum_{i=1}^{\ell} \xi_i - \sum_{i=1}^{\ell} \alpha_i [y_i(w'\phi(x_i) + b) - 1 + \xi_i] - \sum_{i=1}^{\ell} \lambda_i \xi_i \tag{1.11}$$

avec les multiplicateurs de Lagrange $\alpha_i \geq 0$ et $\lambda_i \geq 0$ pour tout $i = 1, ..., \ell$.

En appliquant le théorème de différentiation relative au Lagrangien, nous obtenons :

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w} = 0 & \Rightarrow \quad w = \sum_{j=1}^{\ell} \alpha_j y_j \phi(x_j) \\ \frac{\partial \mathcal{L}}{\partial b} = 0 & \Rightarrow \quad \sum_{j=1}^{\ell} \alpha_j y_j = 0 \\ \frac{\partial \mathcal{L}}{\partial \xi_j} = 0 & \Rightarrow \quad 0 \leq \alpha_j \leq C, \quad \forall j = 1, ..., \ell \end{cases}$$

Nous avons alors :

$$f(x) = sign[\sum_{i=1}^{\ell} \alpha_i y_i k(x_i, x) + b] \tag{1.12}$$

où les α_i sont solutions du problème dual :

$$\max_{\alpha} \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{1.13}$$

$$\text{sous les contraintes :} \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0$$
$$0 \leq \alpha_i \leq C, \quad \text{pour tout } i = 1, ..., \ell$$

Utilisant les conditions de Karush-Kuhn-Tucker (KKT), on démontre que les points x_i tels que $y_i[w'\phi(x_i)+b] > 1$ ont des multiplicateurs α_i nuls. Alors peu de points avec $\alpha_i \neq 0$ participent à la définition du classifieur SVM, $f(x) = sign[\sum_{i=1}^{\ell} \alpha_i y_i k(x_i, x) + b]$. Ce sont ces points qui sont qualifiés de *vecteurs de support* (voir figure 1.5).

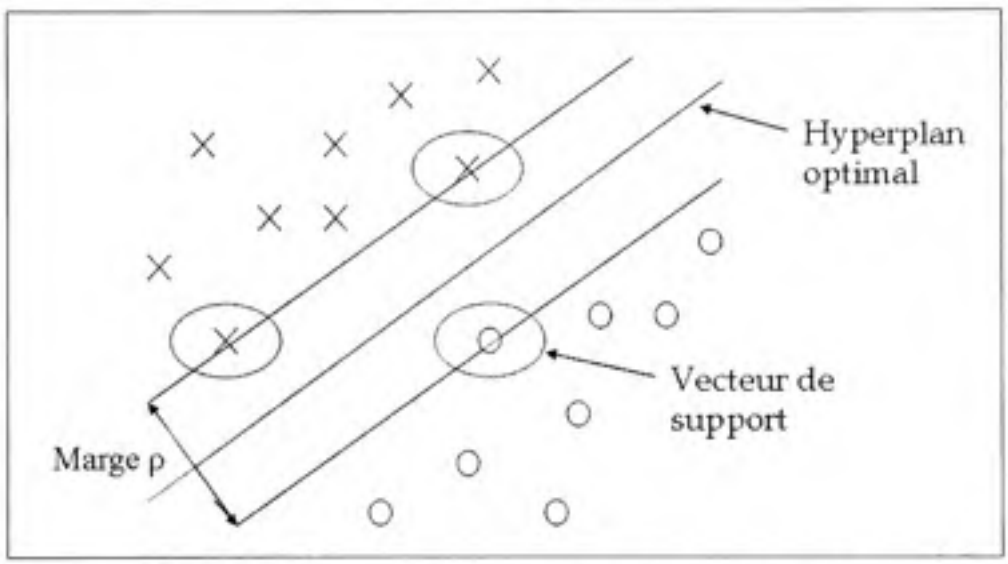


Figure 1.5 Représentation de l'hyperplan et des vecteurs de support.

L'apprentissage en mode supervisé d'une machine à vecteurs de support revient à résoudre un problème de programmation quadratique (QP). 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. [61] permet de faire l'apprentissage des SVMs lorsque la taille de l'ensemble d'apprentissage est large. L'algorithme proposé dans [61] 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 [45] a développé SVM^{light} qui est une technique basée sur le principe de décomposition.

L'algorithme de Joachims [45] diffère de celui d'Osuna et al. [61] par la sélection des éléments de l'ensemble actif. Pour accélérer la convergence, Joachims propose d'utiliser une stratégie basée sur une méthode d'optimisation développée par Zoutendijk [94]. L'idée est de trouver la direction admissible pour la descente de gradient et de former à partir de cette direction l'ensemble actif.

L'algorithme SMO (Sequential Minimal Optimization) développé par Platt [63] 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 . Le SMO, comme proposé dans [63], est une procédure lente. Et ceci dépend surtout des heuristiques utilisées pour sélectionner les deux multiplicateurs à mettre à jour. Dans [51], Keerthi et al. ont proposé une technique de sélection des deux multiplicateurs qui permet une convergence rapide de l'algorithme.

1.3.3 Classification multi-classe avec SVM

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

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

A. 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 couplées pour prendre des décisions lors du test. Le couplage naïf qui consiste à attribuer à une observation la classe dont la sortie de la 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 de la SVM ayant la plus forte valeur de sortie normalisée ou de probabilité. Dans la littérature, nous avons d'autres techniques proposées pour réaliser du multi-classe SVM avec c fonctions discriminantes [15; 38].

B. 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 SVMs convertie en probabilité, fournit pour un exemple x donné :

$$p_{ij} = P(x \in \omega_i / x \in \omega_i \cup \omega_j) \tag{1.14}$$

La règle de décision est alors donnée par :

$$\underset{1 \leq i \leq c}{\operatorname{argmax}} p_i \tag{1.15}$$

avec

$$p_i = \frac{2}{c(c-1)} \sum_{i \neq j} \sigma(p_{ij}), \tag{1.16}$$

où σ désigne la fonction de couplage .

Plusieurs types de couplage sont rapportés dans la littérature [39; 58; 87; 42], mais il demeure toujours un vif débat sur le modèle de couplage le plus efficace. Dans [58], les fonctions de couplage du tableau 1.2 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 dans l'approche "un-contre-tous", il faut utiliser toute la base de données de toutes les classes pour entrainer chaque SVM du système multi-classe.

Tableau 1.2 Fonctions de couplage des SVMs en multi-classe un-contre-un

Fonctions	**Expression**
PWC1	$\sigma(x) = \begin{cases} 1 & \text{if } x > 0.5 \\ 0 & \text{otherwise} \end{cases}$
PWC2	$\sigma(x) = x$
PWC3	$\sigma(x) = \frac{1}{1+exp[-12(x-0.5)]}$
PWC4	$\sigma(x) = \begin{cases} 1 & \text{if } x > 0.5 \\ x & \text{otherwise} \end{cases}$
PWC5	$\sigma(x) = \begin{cases} x & \text{if } x > 0.5 \\ 0 & \text{otherwise} \end{cases}$

1.3.4 Least Squares SVM

Les LS-SVMs (Least Squares Support Vector Machine) [73; 72; 78; 90] constituent une variante des SVMs. Ce sont des machines qui sont aussi basées sur la maximisation de la marge, un principe qui minimise le risque structurel. Les LS-SVMs ont ainsi hérité de l'avantage de généralisation des SVMs. Les modifications apportées au classifieur SVM de Vapnick, permet de résoudre un système linéaire au lieu d'un problème QP lors de l'apprentissage. La nouvelle formulation proposée par Suykens [73] et qui a donné naissance au LS-SVM est la suivante :

$$\min_{w,b,e} \frac{1}{2} w'w + \gamma \frac{1}{2} \sum_{i=1}^{\ell} e_i^2 \tag{1.17}$$

$$\text{sous les contraintes } : y_i[w'\phi(x_i) + b] = 1 - e_i \quad \forall i = 1, ..., \ell \tag{1.18}$$

Nous pouvons remarquer que la formulation de la SVM originale a été modifiée de deux façons. Premièrement, les contraintes d'inégalité sont remplacées par des contraintes d'égalité et deuxièmement une fonction de coût quadratique a été considérée, comme dans le cas des SVMs de norme L2. Ces deux modifications essentielles permettent de simplifier fortement le problème résultant qui donne un système linéaire.

Le Lagrangien du problème (1.17) s'écrit :

$$\mathcal{L}(w, b, e, \alpha) = \frac{1}{2} w'w + \gamma \frac{1}{2} \sum_{i=1}^{\ell} e_i^2 - \sum_{i=1}^{\ell} \alpha_i \{y_i[w'\phi(x_i) + b] - 1 + e_i\} \quad \forall i = 1, ..., \ell \tag{1.19}$$

où α_i sont les multiplicateurs de Lagrange pouvant être positifs ou négatifs à cause des contraintes d'égalité.

Les conditions d'optimalité du problème (conditions de KKT) donnent :

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w} = 0 & \Rightarrow \quad w = \sum_{j=1}^{\ell} \alpha_j y_j \phi(x_j) \\ \frac{\partial \mathcal{L}}{\partial b} = 0 & \Rightarrow \quad \sum_{j=1}^{\ell} \alpha_j y_j = 0 \\ \frac{\partial \mathcal{L}}{\partial e_j} = 0 & \Rightarrow \quad \alpha_j = \gamma e_j, \quad \forall j = 1, ..., \ell \\ \frac{\partial \mathcal{L}}{\partial \alpha_j} = 0 & \Rightarrow \quad y_j[w'\phi(x_j) + b] - 1 + e_j = 0 \quad \forall i = j, ..., \ell \end{cases} \tag{1.20}$$

En posant :

$Z = (y_i\phi(x_1), ..., y_\ell\phi(x_\ell))'$

$Y = (y_1, ..., y_\ell)'$

$\alpha = (\alpha_1, ..., \alpha_\ell)'$

$e = (e_1, ..., e_\ell)'$

$\vec{1} = (1, ..., 1)$

Les égalités de (1.20) deviennent en relation matricielle :

$$w - Z'\alpha = 0 \tag{1.21}$$

$$Y'\alpha = 0 \tag{1.22}$$

$$\gamma e - \alpha = 0 \tag{1.23}$$

$$Zw + bY + e = \vec{1} \tag{1.24}$$

Il apparaît alors que la solution de la LS-SVM est trouvée en résolvant le système linéaire suivant :

$$\begin{pmatrix} 0 & Y' \\ Y & \Omega + \gamma^{-1} I \end{pmatrix} \begin{pmatrix} b \\ \alpha \end{pmatrix} = \begin{pmatrix} 0 \\ \vec{1} \end{pmatrix} \tag{1.25}$$

où $\Omega = Z'Z$ est la matrice carrée dont les éléments sont $\Omega_{ij} = y_i y_j k(x_i, x_j)$.

Pour résoudre numériquement (1.25), on utilise des algorithmes d'optimisation basés sur la technique de gradient conjugué comme proposé par Suykens et al. [73; 72]. Une amélioration de l'algorithme d'apprentissage qui permet d'avoir une réduction d'environ 50% de temps de calcul est développée dans [26]. Nous avons aussi deux autres techniques d'apprentissage

pour les LS-SVMs : la technique du SMO proposée par Keerthi et Shevade [50] et une méthode algébrique développée par Chua dans [27].

En considérant les problèmes d'optimisation (1.5) et (1.17) définissant respectivement la SVM et la LS-SVM, nous remarquons dans les fonctions objectives le terme de maximisation de la marge $2/\|w\|$. Mais avec les contraintes, la marge déterminée par la LS-SVM est plus grande que celle trouvée par la SVM (voir figure 1.6).

En effet, nous avons :

$$\begin{aligned} e_i &= 1 - y_i[w'\phi(x_i) + b] \\ &= [y_i]^2 - y_i[w'\phi(x_i) + b] \\ &= y_i\{y_i - [w'\phi(x_i) + b]\} \end{aligned}$$

Alors minimiser $\sum e_i^2$ est équivalent à minimiser $y_i - [w'\phi(x_i) + b] \quad \forall i = 1, ..., \ell$. Donc, idéalement, il faut avoir $y_i - [w'\phi(x_i) + b] \to 0 \quad \forall i = 1, ..., \ell$, c'est-à-dire $[w'\phi(x_i) + b] \to 1$ pour les données de la classe positive et $[w'\phi(x_i) + b] \to -1$ pour les données de la classe négative. Ainsi, les droites $w'\phi(x_i) + b = 1$ et $w'\phi(x_i) + b = -1$ définissant la marge de séparation sont plus proches du centre de masse des données de chaque classe.

Cependant, la contrainte d'inégalité du SVM $y_i[w'\phi(x) + b] \geq 1 - \xi_i \quad \forall i = 1, ..., \ell$ donne $\xi_i \geq 1 - y_i[w'\phi(x) + b] \quad \forall i = 1, ..., \ell$. Alors, la minimisation de $\xi_i \quad \forall i = 1, ..., \ell$ entraîne qu'il faut que les données soient plus loin possibles de la marge de séparation, d'où la marge de séparation est rétrécie pour contenir peu de données.

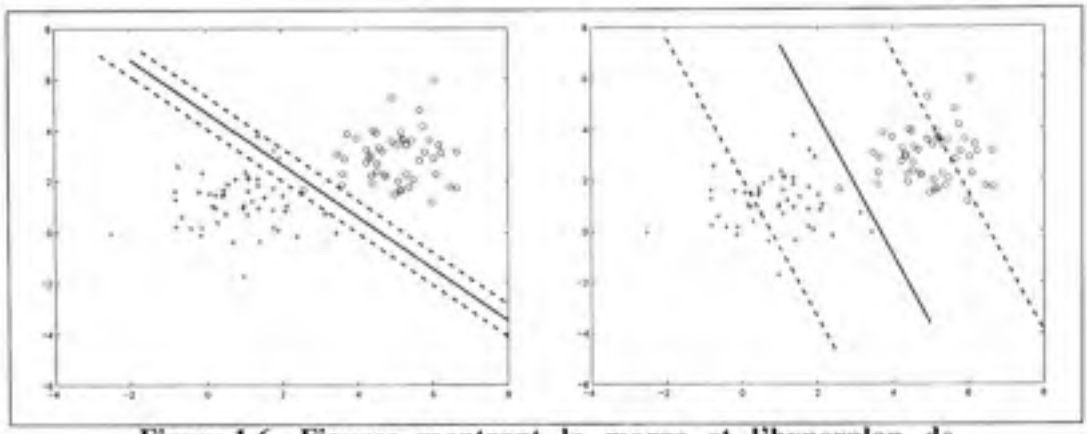


Figure 1.6 Figures montrant la marge et l'hyperplan de séparation trouvés par la SVM (à gauche) et par la LS-SVM (à droite) pour un problème biclasse 2D.

Nous savons que pour définir l'hyperplan de séparation pour une SVM, nous n'avons besoin que de peu de données appelées Vecteurs de Support dont les α_i sont non nuls. Et par les conditions KKT, nous savons aussi que ces points (Vecteurs de Support) sont ceux qui sont mal classés selon la marge. Mais pour la LS-SVM, très peu de points ont $\alpha_i = 0$. Car, pour que α_i soit nul, il faut que le e_i correspondant soit nul. Donc, seuls les points situés sur la marge de séparation ne contribuent pas à la définition de l'hyperplan du classifieur. Ainsi, la machine obtenue par la LS-SVM est plus complexe que celle obtenue par la SVM. Le tableau 1.3 résume certaines différences relevées entre les SVMs et les LS-SVMs.

Pour résoudre ce problème qui limite l'utilisation des classifieurs LS-SVMs dans des applications réelles, nous pouvons suggérer l'entraînement des LS-SVMs dans l'espace primal en utilisant la stratégie KPCA (Kernel Principal Component Analysis). Cependant, dans la littérature, certaines méthodes ont été proposées pour réduire la complexité des LS-SVMs, c'est-à-dire le nombre de vecteurs de support à retenir après apprentissage de la machine[71; 18; 31; 89; 44].

Suykens et al. [71] proposent d'éliminer les exemples d'apprentissage dont la valeur des α_i correspondants est très petite, inférieure à un seuil fixée. Dans [31], une méthode d'élimination

plus sophistiquée a été proposée en utilisant une approximation d'erreur. Cependant, ces deux méthodes ne permettent pas d'éliminer les "outliers" qui ont une influence négative sur le classifier entraîné avec une base d'apprentissage réduite. Pour cela, Li et al. [89] propose une autre technique d'élimination améliorée en utilisant certaines heuristiques.

Dans [18; 19], une autre façon de réduire la complexité des LS-SVMs a été proposée. Les auteurs suggèrent de présélectionner un sous-ensemble d'exemples d'apprentissage qui sont pertinents et de les utiliser pour faire l'apprentissage des LS-SVMs. Enfin, nous pouvons citer le travail de Jia and al. [44] qui ont proposé "Fast Sparse Approximation for LS-SVM" (FSALS-SVM) qui est un algorithme rapide donnant un classifieur de bonne performance de généralisation et avec un petit nombre de vecteurs de support.

Tableau 1.3 Tableau comparatif entre la SVM et la LS-SVM

	SVM	**LS-SVM**
Marge	Rétrécie et contenant peu de données	Large et contenant beaucoup de données
Apprentissage	programmation quadratique	système linéaire
Phase de test	complexe	très complexe

1.3.5 Transductive SVM

En général, l'apprentissage automatique utilise l'inférence inductive pour estimer la valeur d'une fonction en un point donné en deux étapes. D'abord, les paramètres du modèle représentant la fonction sont estimés pour tout l'espace d'entrée considéré en utilisant des exemples

d'apprentissage. Puis, pour chaque exemple de test donné, on calcule la valeur de la fonction avec les paramètres estimés préalablement. Cependant, il est possible de trouver la valeur de la fonction en un point de test en une étape. Cette façon directe d'opérer s'appelle inférence transductive [83; 82]. Et puisque au cours de l'apprentissage, le but final est d'estimer les valeurs de la fonction en des points spécifiques et non de façon générale pour tout l'espace, cette approche s'est révélée plus exacte que l'inférence inductive et moins difficile.

La Transductive SVM (TSVM) est une variante intéressante des SVMs utilisant l'approche transductive. Dans ce cas-ci, on cherche l'hyperplan de séparation $< w, b >$ et les étiquettes $y_1^*, ..., y_n^*$ probables des données non étiquetées (données de test spécifiques) qui maximisent la marge et minimisent l'erreur. Ainsi, en classification les étiquettes sont obtenues en une seule étape :

$$\min_{w,b,\xi,\xi^*,y_1^*,...,y_n^*} \frac{1}{2}w'w + C\sum_{i=1}^{\ell}\xi_i + C^*\sum_{j=1}^{n}\xi_j^* \tag{1.26}$$

$$\text{sous les contraintes} : y_i[w'\phi(x_i) + b] \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i = 1, ..., \ell \tag{1.27}$$

$$y_j^*[w'\phi(x_j^*) + b] \geq 1 - \xi_j^*, \quad \xi_j^* \geq 0 \quad \forall j = 1, ..., n \tag{1.28}$$

où $x_1^*, ..., x_n^*$ désignent les données non étiquetées et $y_j^* \in \{-1; 1\}$.

Vapnik [83] a proposé cette formulation afin de renforcer la précision de la classification sur des données spécifiques connues d'avance en ajoutant la minimisation de l'erreur sur cette base de test $\{x_1^*, \ldots, x_n^*\}$. Ainsi, la TSVM maximise la marge en utilisant des données étiquetées et non étiquetées en plus de la minimisation de l'erreur sur toute l'ensemble des données. Elle implémente alors l'hypothèse de regroupement (cluster assumption) pour l'apprentissage semi-supervisé et dépasse l'idée stricte de la transduction [25]. De cette manière, la formulation de la TSVM va au delà de l'idée première de Vapnik qui est la transduction, elle permet de faire de la SVM semi-supervisée (S^3VM) [22].

La formulation de la $TSVM/S^3VM$ conduit à un problème non-convexe, ce qui rend difficile son application. Dans la littérature, nous avons recensé certains algorithmes [48; 10; 32; 35; 22] qui ont traité de la résolution du problème d'optimisation qui peut est réduit comme suit :

$$\min_{w,b,Y^\star} \mathcal{I}(w,b,Y^\star) = \frac{1}{2}\|w\| + C\sum_{i=1}^{\ell} H(y_i,o_i) + C^\star \sum_{j=1}^{n} H(y_j^\star,o_j^\star) \tag{1.29}$$

$$\tag{1.30}$$

où $o_i = w'\phi(x_i) + b$ et H désigne la fonction perte "Hinge Loss".

Souvent pour résoudre ce problème, une contrainte supplémentaire est utilisée afin d'éliminer les solutions non équilibrées. Ce sont des solutions qui pouvaient donner $y_j^\star = 1, \forall j$ ou $y_j^\star = -1, \forall j$, c'est-à-dire, toutes les données non étiquetées sont classées dans la même classe. Ce phénomène se produit lorsque la taille de l'ensemble des données non étiquetées est très large par rapport à celle des données étiquetées. La contrainte utilisée dans la plupart des implémentations de $TSVM/S^3VM$ est la suivante :

$$R = \frac{1}{n}\sum_{j=1}^{n} max(y_j^\star, 0) = \frac{1}{2}\left[1 + \frac{1}{n}\sum_{j=1}^{n} y_j^\star\right] \tag{1.31}$$

où R est estimé à partir des connaissances *a priori* ou à partir de l'ensemble des données étiquetées.

1.3.5.1 SVM-light

En 1999, Joachims [48] a développé pour la première fois l'algorithme d'apprentissage pour la TSVM en utilisant la technique d'auto-apprentissage combinée avec des heuristiques. Au cours de l'auto-étiquetage, l'algorithme se base sur la contrainte exprimée par 1.31 pour étiqueter une fraction R des données positives. Et puis de façon itérative, deux exemples opposés peut changer d'étiquette lorsque cela permet de réduire la fonction objective. Cet algorithme a

été appliqué au problème de classification de textes avec succès. D'autres variantes de l'algorithme de Joachims avec des améliorations sont proposées dans [88; 86].

1.3.5.2 Mixed-Integer

Une année plutôt, Bennett et al. [10] avaient proposé une nouvelle formulation du problème d'apprentissage semi-supervisé pour les SVM afin d'en déduire une implémentation efficace. Dans cette formulation, l'erreur de classification des données non étiquetées est calculée pour le cas où les données appartiennent à la classe -1 et à la classe +1 et on prend le minimum pour évaluer la fonction objective (1.26). Le problème d'optimisation est alors défini comme suit :

$$\min_{w,b,\xi,z,t} \frac{1}{2} w'w + C\left[\sum_{i=1}^{\ell} \xi_i + \sum_{j=1}^{n} min(t_j, z_j)\right] \tag{1.32}$$

$$\text{sous les contraintes} : y_i[w'\phi(x_i) + b] \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i = 1, ..., \ell \tag{1.33}$$

$$+[w'\phi(x_j^*) + b] + t_j \geq 1, \quad t_j \geq 0 \quad \forall j = 1, ..., n \tag{1.34}$$

$$-[w'\phi(x_j^*) + b] + z_j \geq 1, \quad z_j \geq 0 \quad \forall j = 1, ..., n \tag{1.35}$$

Pour résoudre ce problème d'optimisation, Bennett et al. [10] ont utilisé la méthode « Interger Programming ».

Ils ont proposé d'ajouter une variable supplémentaire au problème, d_j appelée variable de décision : $d_j = 1$ pour désigner la classe positive et $d_j = 0$ la classe négative. Le nouveau problème à résoudre se présente comme suit, où nous avons des variables réelles et entières :

$$\min_{w,b,\xi,z,t,d} \frac{1}{2} w'w + C\left[\sum_{i=1}^{\ell} \xi_i + \sum_{j=1}^{n} (t_j + z_j)\right] \tag{1.36}$$

$$\text{sous les contraintes} : y_i[w'\phi(x_i) + b] \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i = 1, ..., \ell \tag{1.37}$$

$$+[w'\phi(x_j^*) + b] + t_j + M(1 - d_j) \geq 1, \quad t_j \geq 0 \quad \forall j = 1, ..., n \tag{1.38}$$

$$-[w'\phi(x_j^*) + b] + z_j + Md_j \geq 1, \quad z_j \geq 0 \quad \forall j = 1, ..., n \tag{1.39}$$

$$d_j = \{0, 1\} \quad \forall j = 1, ..., n \tag{1.40}$$

M est une constante strictement positive de valeur très large pour que si $d_j = 0$ alors $t_j = 0$ et si $d_j = 1$ alors $z_j = 0$. Le précédent problème est résolu en utilisant un programme comme CPLEX.

1.3.5.3 Deterministic Annealing

"Deterministic Annealing" est une autre technique proposée pour la S^3VM. Elle ressemble à la technique décrite dans la section précédente. Le problème 1.29 a été reformulée en introduisant cette fois des variables continues $p = (p_1, ..., p_n)$ où p est vue comme la probabilité d'avoir $y_j^* = 1$:

$$\min_{w,b,p} \mathcal{J}(w, b, p) = \frac{1}{2} \|w\| + C \sum_{i=1}^{\ell} H(y_i, o_i) + C^* \sum_{j=1}^{n} [p_j H(1, o_j^*) + (1 - p_i) H(-1, o_j^*)] + \gamma E(p) \tag{1.41}$$

où $E(p) = \sum_{j=1}^{n} p_j log(p_j) + (1 - p_j) log(1 - p_j)$ désigne un terme de régularisation (entropy term).

De manière alternative, en gardant p fixe, la minimisation de $\mathcal{J}(w, b, p)$ se fait sur les autres variables et on obtient (w_0, b_0). Puis avec (w, b) fixe, $\mathcal{J}(w_0, b_0, p)$ est minimisée à nouveau, mais cette fois-ci pour obtenir la valeur optimale de p. Et le processus est répétée jusqu'à ce que la valeur de l'entropie atteigne un seuil fixé au préalable [69].

1.3.5.4 Concave Convex Procedure (CCCP)

La méthode d'optimisation $CCCP$ est appliquée au S^3VM par Collobert et al.[28]. Il s'agit de décomposer la fonction non-convexe en une composante convexe et une autre concave. Et à chaque itération, la partie concave est remplacée par une approximation tangentielle (développement à l'ordre 1 de Taylor). Dans le cas des S^3VMs, les deux premiers termes de $\mathcal{J}(w, b, p)$ sont convexes. C'est donc le dernier terme exprimant l'erreur sur les données non-étiquetées qui est décomposer en deux :

$$H(1, o_j^*) = max(0, 1 - |o_j^*|^2) = \underbrace{max(0, 1 - |o_j^*|^2) + 2|o_j^*|}_{convexe} \underbrace{-2|o_j^*|}_{concave} \tag{1.42}$$

en considérant la fonction "Hinge Loss" au carrée.

1.3.5.5 Autres méthodes

Nous avons aussi la méthode "Branch-and-Bound" qui permet de faire une optimisation globale mais très coûteux en temps et impraticable pour les larges bases de données. Chapelle et al.[21] ont appliqué cette méthode pour obtenir des solutions globales sur des petites bases de données. Enfin, nous pouvons citer la technique de relaxation [11], de descente de gradient [25] et méthode de Newton [22] qui sont utilisées pour optimiser $\mathcal{J}(w, b, p)$ après avoir reformulé de problème.

Malgré toutes ces panoplies d'algorithmes, le problème de convexité hante toujours les solutions trouvées qui ne sont pas toujours optimales. Car, les algorithmes proposés convergent parfois vers des minimums locaux.

CHAPITRE 2

FORMULATION BAYÉSIENNE POUR L'APPRENTISSAGE SEMI-SUPERVISÉ

2.1 Introduction

L'inférence bayésienne permet d'intégrer dans l'estimation du modèle une connaissance *a priori* sur ce dernier. C'est une approche qui s'adapte convenablement aux problèmes où la connaissance est partielle ou probabiliste [54]. Dans la théorie bayésienne, la connaissance *a priori* est tout d'abord exprimée sous forme de probabilité. Puis avec l'arrivée des données, cette connaissance est mise à jour à l'aide des nouvelles informations tirées de ces données. Ainsi, la connaissance *a priori* est raffinée pour donner la connaissance *a postériori* qui est plus consiste.

Considérons un cadre de travail bayésien où nous cherchons à estimer le paramètre θ du modèle d'un classifieur ainsi que les étiquettes des données non étiquetées que nous désignons par $z = (y_1^\star, y_2^\star, ..., y_n^\star)$. Dans ce chapitre, nous allons utiliser l'approche bayésienne basée sur un et deux niveaux d'inférence pour déterminer le paramètre du modèle et les étiquettes. Par suite, avec cette méthodologie, nous allons déduire des algorithmes d'apprentissage en mode semi-supervisé pour les SVMs et les LS-SVMs.

2.2 Estimation avec un seul niveau d'inférence : Maximum a posteriori(MAP)

Il s'agit de déterminer le paramètre θ du modèle et les étiquettes z des données non étiquetées en utilisant un seul niveau d'inférence. Le théorème de Bayes permet de relier les distributions *a priori* et *a posteriori* sur le paramètre et les étiquettes :

$$P(\theta, z|\mathcal{D}) = \frac{P(\mathcal{D}, \theta, z)}{P(\mathcal{D})} \tag{2.1}$$

où $P(\mathcal{D}, \theta, z)$ est la probabilité jointe des données
$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), ..., (x_\ell, y_\ell), x_1^\star, x_2^\star, ..., x_n^\star\}$, du paramètre θ du modèle et des étiquettes z.

En utilisant la distribution *a priori* du paramètre θ, le postérieur devient :

$$P(\theta, z|\mathcal{D}) = \frac{P(\mathcal{D}, z|\theta)}{P(\mathcal{D})} P(\theta) \tag{2.2}$$

Il reste alors à maximiser le postérieur pour déterminer le paramètre θ du modèle et les étiquettes z des données sans étiquettes. En général, puisque le dénominateur ne dépend pas des paramètres, on maximise le logarithme du numérateur, qui s'écrit :

$$\mathcal{L}(\theta, z) = log[P(\mathcal{D}, z|\theta)P(\theta)] \tag{2.3}$$

La question la plus importante est de savoir comment procéder à l'optimisation de cette expression où les espaces de recherche ne sont pas les mêmes, continu [1] pour le paramètre du modèle et fortement discontinu [2] pour les étiquettes z des données sans étiquettes.

2.3 Estimation avec deux niveaux d'inférence

Le processus d'inférence est décomposée en deux étapes où le paramètre du modèle et les étiquettes sont découplés afin d'être estimés à chaque niveau.

Au premier niveau, l'inférence du paramètre du modèle θ est réalisée avec le postérieur qui s'écrit :

1. Mais cela dépend de la densité a priori de θ
2. Par exemple en classification binaire les éléments de z sont -1/1 ou 0/1

$$P(\theta|\mathcal{D}, z) = \frac{P(\mathcal{D}, z|\theta)}{P(\mathcal{D}, z)} P(\theta) \tag{2.4}$$

où $P(\mathcal{D}, z|\theta)$ est la probabilité conditionnelle des données $\mathcal{D}$ et des étiquettes z sachant le modèle θ , et $P(\theta)$ est la probabilité *a priori* du modèle. La probabilité jointe $P(\mathcal{D}, z)$ est obtenu en intégrant le numérateur de (2.4) sur toute l'espace de définition du paramètre θ :

$$P(\mathcal{D}, z) = \int_{\Theta} P(\mathcal{D}, z|\theta) P(\theta) d\theta \tag{2.5}$$

Au deuxième niveau, l'inférence des étiquettes z est procédée comme suit :

$$P(z|\mathcal{D}) = \frac{P(\mathcal{D}, z)}{P(\mathcal{D})} \tag{2.6}$$

La probabilité jointe utilisée pour exprimer le postérieur de l'équation (2.6) se calcule en se servant de l'équation (2.5), représentant l'évidence dans la formule de Bayes au premier niveau d'inférence. Ainsi, l'équation (2.6) devient :

$$P(z|\mathcal{D}) \propto P(\mathcal{D}, z) \tag{2.7}$$

$$\propto \int_{\Theta} P(\mathcal{D}, z|\theta) P(\theta) d\theta \tag{2.8}$$

L'apprentissage consiste donc à chercher z qui maximise l'équation (2.7) et ensuite utiliser cette valeur dans l'équation (2.4) pour estimer le paramètre θ. La difficulté de calcul dans cette formulation est l'évaluation de l'intégrale de l'équation (2.5) qui permet d'éliminer le paramètre θ.

2.4 SVM semi-supervisé avec un seul niveau d'inférence

Cette section décrit l'application de l'inférence à un niveau aux SVMs dans le contexte de l'apprentissage semi-supervisé. Utilisant l'interprétation bayésienne pour les SVMs décrite dans [70] où la distribution à priori du modèle suit une loi gaussienne (la maximisation de la marge) et la fonction de perte de la SVM est interprétée comme la vraisemblance.

$$P(\mathcal{D}, z|\theta) \propto \prod_i Q(y_i|x_i, \theta)Q(x_i) = \prod_i exp\big(-Cl(y_i[w'\phi(x_i) + b])\big)Q(x_i) \tag{2.9}$$

et

$$P(\theta) \propto exp\big(-\frac{1}{2}\|w\|^2\big) \tag{2.10}$$

avec $\theta = (w, b)$ et $l(t) = \max(0, 1 - t)$ est la fonction perte des SVMs appelée "hinge loss".

En utilisant la précédente formulation bayésienne pour les SVMs, nous obtenons :

$$\mathcal{L}(\theta, z) = \mathcal{L}(w, b, z) = -\frac{1}{2}\|w\|^2 - C\sum l(y_i[w'\phi(x_i) + b]) \tag{2.11}$$

Il s'agit donc de minimiser $-\mathcal{L}(w, b, z)$, ce qui conduit à :

$$\min_{w,b,y_1^*,\ldots,y_n^*} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell} l(y_i[w'\phi(x_i) + b]) + C^*\sum_{j=1}^{n} l(y_j^*[w'\phi(x_j^*) + b]) \tag{2.12}$$

Le problème formulé en (2.12) conduit au problème d'optimisation proposé par Vapnik [83] dans le cadre du TSVM. Et comme décrite plutôt, cette formulation permet de réaliser l'apprentissage semi-supervisé des SVMs.

Dans certains cas, la minimisation de $-\mathcal{L}(w, b, z)$ ne donne pas toujours une bonne solution, alors certaines contraintes sont utilisées sur la proportion des étiquettes à déterminer. En ce qui nous concerne, nous proposons de renforcer la régularisation du classifieur en utilisant le nombre de vecteurs de support.

Ainsi, nous proposons de minimiser l'expression suivante :

$$\mathcal{L}_0(w, b, z) = \beta \left[\frac{1}{2}\|w\|^2 + C \sum l(y_i[w'\phi(x_i) + b]) \right] + (1 - \beta)T(z) \qquad (2.13)$$

où $0 < \beta < 1$ et $T(z)$ représente le nombre de vecteurs de support de la machine obtenu en assignant des étiquettes spécifiques à z.

Nous remarquons que la formulation standard de S^3VM est modifiée en ajoutant un terme qui pénalise la complexité du classifieur en incorporant la réduction du nombre de vecteurs de support dans la fonction objective. Cependant, ce nouveau terme rend plus difficile le problème d'optimisation.

Nous proposons alors d'utiliser les algorithmes génétiques (AGs) [40; 41] pour solutionner le problème de minimisation de la quantité exprimée en 2.13. Les algorithmes génétiques ont l'avantage de s'adapter à des problèmes complexes où la dérivabilité ou la convexité ne sont pas nécessaires. Aussi, la continuité de l'espace de recherche n'est pas une contrainte pour les AGs, comme dans notre cas, les étiquettes des exemples non étiquetés appartiennent à un ensemble discret. De plus, les AGs font de la recherche parallèle contrairement aux algorithmes classiques d'optimisation, ce qui permet d'explorer de vaste espace de recherche.

La stratégie que nous avons proposé pour réaliser l'apprentissage est représenté en figure 2.1. Nous utilisons un algorithme d'apprentissage standard pour entraîner les SVMs en mode supervisé et AG pour attribuer les étiquettes appropriées aux données sans étiquettes. Les détails de l'implémentation de cette technique se retrouvent dans le premier article en annexe, intitulé "*Genetic Algorithm based training for Semi-supervised SVM*".

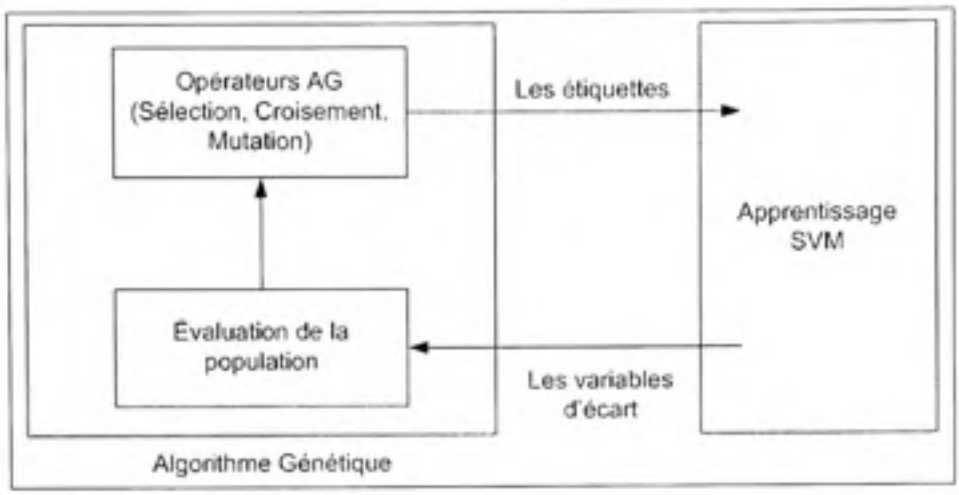


Figure 2.1 Représentation schématique de l'algorithme $S^3VM - GA$.

2.5 LS-SVM semi-supervisé avec un seul niveau d'inférence

En considérant le développement fait dans la section précédente, la LS-SVM semi-supervisé avec un seul niveau d'inférence ($S^2LS - SVM$) se formule comme suit :

$$\min_{w,b,e,e^\star,y_1^\star,\ldots,y_n^\star} \frac{1}{2}w'w + \frac{1}{2}\gamma\sum_{i=1}^{\ell} e_i^2 + \frac{1}{2}\gamma^\star\sum_{j=1}^{n}(e_j^\star)^2 \tag{2.14}$$

$$\text{s.t.} : e_i = y_i - [w'\phi(x_i) + b], \quad \forall i = 1,\ldots,\ell \tag{2.15}$$

$$e_j = y_j^\star - [w'\phi(x_j^\star) + b], \quad \forall j = 1,\ldots,n \tag{2.16}$$

$$y_j^\star \in \{-1,1\} \quad \forall j = 1,\ldots,n \tag{2.17}$$

Pour réaliser l'apprentissage de $S^2LS - SVM$, nous avons proposé deux algorithmes qui sont détaillés dans le deuxième article en annexe, intitulé *"Semi-Supervised Least Squares Support Vector Machine"*.

Le premier algorithme que nous dénommons $S^2LS - SVM1$ est inspiré de la stratégie utilisée par Joachims pour la TSVM [48]. Tout d'abord, nous commençons par entraîner le classifieur avec les données étiquetées et avec la fonction de décision temporaire, nous procédons à l'éti-

quetage des autres données en se basant sur le ratio entre les exemples positifs et négatifs. De plus, au cours des autres itérations, les étiquettes des données mal classées selon la marge c'est-à-dire $y^\star\alpha^\star > 0$ et dont les valeurs $e_j^\star = \alpha_j^\star/\gamma^\star$ correspondantes sont très larges, sont changées afin de minimiser la fonction objective. L'algorithme 1 montre les grandes étapes de ce premier algorithme.

Algorithme 1 Apprentissage de la LS-SVM semi-supervisé

Entrée Les exemples étiquetés $\{X, Y\}$ et les exemples non étiquetés $X^\star$

Sortie Paramètres du classifieur $< w, b >$

- Entraîner la machine avec les exemples étiquetées
- Calculer la sortie pour chaque exemple sans étiquette
- Étiqueter ces exemples en utilisant le ratio

tantque La fonction objective décroit **faire**

  Entraîner la machine avec tous les exemples

  Sélectionner l'exemple sans étiquette dont $|\alpha^\star|$ correspondant est le plus large

  si $y^\star\alpha^\star > 0$ **alors**

    Changer l'étiquette en son opposé

  finsi

fin tantque

Le deuxième algorithme que nous avons proposé, permet d'étiqueter graduellement les exemples d'apprentissage non étiquetés. À chaque itération, l'exemple qui permet d'avoir la plus petite variation de la fonction objective est sélectionné et ajouté aux données étiquetées. Le critère de sélection est basé sur la valeur de $\alpha^\star$ qui vaut $a_j^{(1)}$ lorsque l'étiquette à attribuer est $y_j^\star = 1$ et $a_j^{(-1)}$ dans le cas contraire. Les détails de cette seconde approche sont montrés par l'algorithme 2.

2.6 LS-SVM semi-supervisé avec deux niveaux d'inférence

Il est difficile d'appliquer le modèle d'inférence à deux niveaux à la SVM originale. Dans ce qui suit, je présente son application au LS-SVM dont la formulation bayésienne permet de

Algorithme 2 Apprentissage progressif pour la LS-SVM semi-supervisé

Entrée Les exemples étiquetés $\{X, Y\}$ et les exemples non étiquetés X^*

Sortie Paramètres du classifieur $< w, b >$

Entraîner la machine avec les exemples étiquetées

tantque Des exemples sans étiquette restent **faire**

- Sélectionner le signe de la prochaine étiquette selon la fonction objective
- Calculer $a_j^{(1)}$ et $a_j^{(-1)}$ pour les exemples sans étiquette restants

si l'étiquette est positive **alors**

Trouver l'exemple dont le $a_j^{(1)}$ correspondant est le plus petit et attribuer l'étiquette 1

sinon

Trouver l'exemple dont le $a_j^{(-1)}$ correspondant est le plus petit et attribuer l'étiquette -1

finsi

fin tantque

Entraîner la machine finale avec tous les exemples

calculer aisément l'intégrale de l'équation (2.5).

Les équations donnant la formule de la vraisemblance et celle de la distribution *a priori* du paramètre θ sont exprimées selon le travail de Suykens et al. [73; 72; 78; 90] :

$$P(\mathcal{D}, z|\theta) = \prod_i \sqrt{\beta/2\pi} exp\big(-\frac{1}{2}\beta e_i^2\big) \times \prod_i Q(x_i) \tag{2.18}$$

avec

$$e_i = y_i - [w'\phi(x_i) + b] \tag{2.19}$$

et

$$P(\theta) = P(w,b) = \Big(\frac{\mu}{2\pi}\Big)^{n_h/2} exp\big(-\frac{1}{2}\mu w'w\big)\frac{1}{\sqrt{2\pi}\sigma_b} exp\big(-\frac{1}{2\sigma_b^2}b^2\big) \tag{2.20}$$

où n_h désigne la dimension de l'espace augmentée pour la fonction de projection ϕ.

La distribution *a priori* sur le biais b approxime une distribution uniforme lorsque $\sigma_b \longrightarrow \infty$ et on obtient donc une forme plus simplifiée pour la densité a priori sur $\theta = (w,b)$.

$$P(w,b) = \Big(\frac{\mu}{2\pi}\Big)^{n_h/2} exp\big(-\frac{1}{2}\mu w'w\big) \tag{2.21}$$

Pour simplification, nous posons :

$$\tilde{\phi}(x_i) = (\phi(x_i), 1) \tag{2.22}$$

et

$$J = \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} \tag{2.23}$$

où I est la matrice identité.

Alors les équations de la vraisemblance et la densité *a priori* deviennent :

$$P(\mathcal{D}, z|\theta) = \prod_i \sqrt{\beta/2\pi} exp\big(-\frac{1}{2}\beta(y_i - \theta'\tilde{\phi}(x_i))^2\big) \times \prod_i Q(x_i) \tag{2.24}$$

et

$$P(\theta) = \Big(\frac{\mu}{2\pi}\Big)^{n_h/2} exp\big(-\frac{1}{2}\mu\theta' J\theta\big) \tag{2.25}$$

Par suite, nous avons :

$$\begin{aligned} P(\mathcal{D}, z|\theta)P(\theta) &= \prod_i \sqrt{\beta/2\pi} exp\big(-\frac{1}{2}\beta(y_i - \theta'\tilde{\phi}(x_i))^2\big) \\ &\times \Big(\frac{\mu}{2\pi}\Big)^{n_h/2} exp\big(-\frac{1}{2}\mu\theta' J\theta\big) \prod_i Q(x_i) \\ &= \Big(\frac{\beta}{2\pi}\Big)^{(\ell+n)/2} \Big(\frac{\mu}{2\pi}\Big)^{n_h/2} exp\big(-\frac{1}{2}\mu\theta' J\theta \\ &- \frac{1}{2}\beta\sum_i (y_i - \theta'\tilde{\phi}(x_i))^2\big) \prod_i Q(x_i) \\ &= \Big(\frac{\beta}{2\pi}\Big)^{(\ell+n)/2} \Big(\frac{\mu}{2\pi}\Big)^{n_h/2} exp\Big(-\frac{1}{2}\mu\theta' J\theta \\ &- \frac{1}{2}\beta\sum_i [y_i^2 - 2y_i\theta'\tilde{\phi}(x_i) + \theta'(\tilde{\phi}(x_i)\tilde{\phi}(x_i)')\theta]\Big) \prod_i Q(x_i) \end{aligned}$$

Ce qui donne :

$$P(\mathcal{D}, z|\theta)P(\theta) = \Lambda exp\Big[-\frac{1}{2}(\theta - \theta_{MAP})' H(\theta - \theta_{MAP})\Big] exp\Big[\frac{1}{2}F_y' H^{-1} F_y\Big] \times \prod_i Q(x_i) \tag{2.26}$$

où

$$\Lambda = \Big(\frac{\beta}{2\pi}\Big)^{(\ell+n)/2} \Big(\frac{\mu}{2\pi}\Big)^{n_h/2} exp\Big(-\frac{1}{2}\beta(\ell+n)\Big) \tag{2.27}$$

$$F_y = \beta\sum_i y_i\tilde{\phi}(x_i) = \beta\sum_{i=1}^{\ell} y_i\tilde{\phi}(x_i) + \beta\sum_{j=1}^{n} y_j^*\tilde{\phi}(x_j^*) \tag{2.28}$$

$$H = \beta\sum_i (\tilde{\phi}(x_i)\tilde{\phi}(x_i)') + \mu J \tag{2.29}$$

$$\theta_{MAP} = H^{-1}F_y \tag{2.30}$$

Alors l'intégrale (2.5) vaut :

$$\int P(\mathcal{D}|\theta, z)P(\theta)d\theta = \Lambda\sqrt{(2\pi)^{n_h+1}detH^{-1}}exp\Big[\frac{1}{2}F_y'H^{-1}F_y\Big] \times \prod_i Q(x_i) \tag{2.31}$$

Par conséquent, nous avons :

$$P(z|\mathcal{D}) \propto exp\Big[\frac{1}{2}F_y'H^{-1}F_y\Big] \tag{2.32}$$

Il faut donc maximiser $F_y'H^{-1}F_y$, c'est-à-dire trouver les étiquettes $y_1^*, ..., y_n^*$ pour que l'expression $F_y'H^{-1}F_y$ soit maximale. Nous avons donc à résoudre un problème d'optimisation combinatoire. L'algorithme d'apprentissage qui s'en déduit est montré en algorithme 3, où le modèle obtenu est appelé *"Bayesian semi-supervised least-square SVM"* ($BS^2LS - SVM$). Le troisième article référé en annexe donne plus de détails sur cette méthode.

Algorithme 3 LS-SVM semi-supervisé avec deux niveaux d'inférence : $BS^2LS - SVM$

Entrée Exemples étiquetées $\{X, Y\}$ et Exemples sans étiquettes X^*

Sortie Hyperplan $< w, b >$

- Sélectionner la fonction noyau et ses paramètres
- Fixer les hyperparamètres
- Entraîner la machine avec les exemples étiquetées $\{X, Y\}$
- Étiqueter les exemples sans étiquette qui sont classifiés avec confiance
- Trouver les étiquettes qui maximisent (2.32)
- Déterminer le paramètre du modèle en utilisant (2.30)
- Retourner les paramètres du classifieur $< w, b >$

CHAPITRE 3

APPRENTISSAGE SEMI-SUPERVISÉ SOUTENU (HELP-TRAINING)

3.1 Introduction

L'une des techniques classiques utilisées en apprentissage semi-supervisé est l'auto- apprentissage (self-training) que nous avons décrit dans le chapitre 1. La première étape de cette méthode consiste à entraîner le classifieur en supervisé avec les données étiquetées. Puis avec cette machine temporaire qui a souvent une très faible capacité de généralisation, les données sans étiquettes sont de façon incrémentale étiquetées. À chaque itération, les données sans étiquettes classées avec plus de confiance sont ajoutées à l'ensemble d'apprentissage avec leurs étiquettes prédites et sont utilisées pour accroître le pouvoir de généralisation du classifieur à l'itération suivante. Cependant, cette technique ne fonctionne pas assez avec les classifieurs discriminants. Prenons l'exemple de la SVM évoluant dans le processus d'auto-apprentissage (Figure 3.1). Les exemples x1 and x2 seront mal classés avec plus de confiance que les autres, ce qui va produire une frontière de décision erronée à la fin du processus. Pour remédier à cette erreur d'appréciation de la confiance des classifieurs discriminants, nous proposons d'utiliser un classifieur par modélisation pour l'aider dans la prise de décision. Nous appelons cette technique *Apprentissage semi-supervisé soutenu (Help-Training)* [4].

3.2 Classifieurs par modélisation versus par séparation

La mise en œuvre d'un classifieur est équivalent à approximer une fonction inconnue $f : X \mapsto Y$ ou estimer une probabilité $P(Y|X)$, où X représente l'entrée et Y les étiquettes. Il existe deux grandes familles de méthodes pour estimer les paramètres de la fonction f : méthode par modélisation (modèle génératif) et méthode par séparation (modèle discriminant).

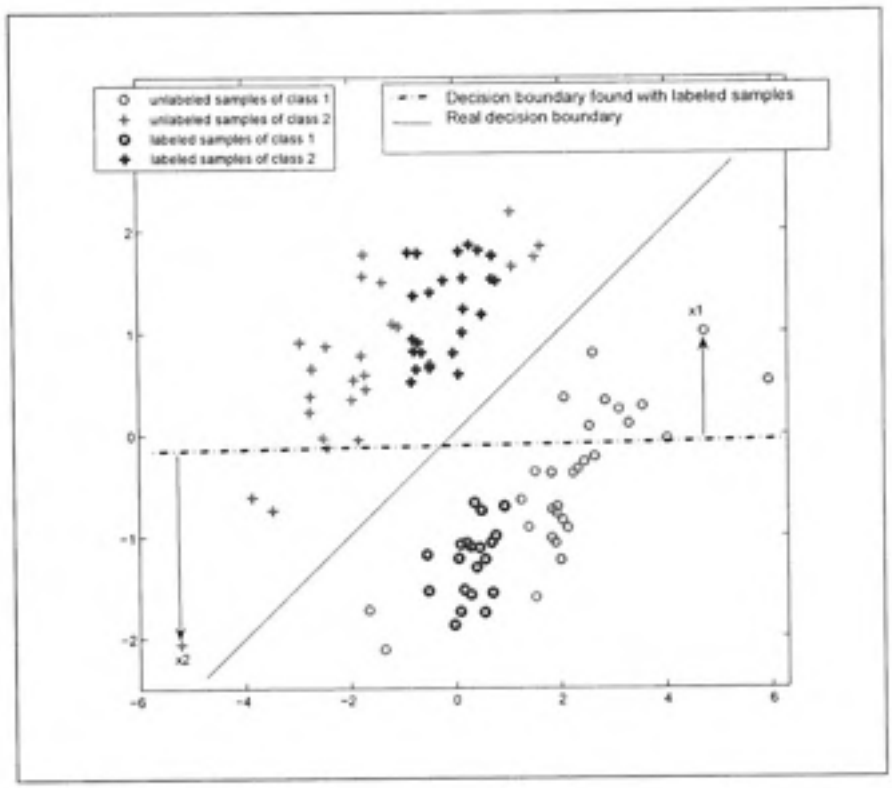


Figure 3.1 Illustration du mauvais fonctionnement de l'auto-apprentissage avec les classifieurs discriminants comme la SVM. Les exemples x1 and x2 seront mal classés avec plus de confiance que les autres, ce qui va biaiser la frontière de décision à l'itération suivante.

La méthode par modélisation consiste à utiliser les données d'apprentissage pour estimer les probabilités $P(X|Y)$ et $P(Y)$. Puis, la probabilité $P(Y|X)$ est déterminée en se servant de la formule de Bayes :

$$P(Y|X) = \frac{P(X|Y)P(Y)}{\sum_y P(X|y)P(y)} \tag{3.1}$$

L'estimation de la distribution $P(X|Y)$ est la modélisation de chaque classe et consiste à estimer la distribution des données issues de chaque classe, c'est-à-dire déterminer comment les

données de chaque classe sont générées. D'où on parle d'approche génératif ou de reproduction.

La deuxième façon de construire un classifieur est d'estimer directement la probabilité $P(Y|X)$ ou $Y = f(X)$. Avec ce type de méthode, on cherche à trouver les paramètres de f en utilisant la séparation des classes. Chaque classe doit être explicitement séparée des autres. On parle alors de classifieur discriminant.

Ces deux principales méthodes utilisées pour concevoir des classifieurs, sont très utiles et complémentaires, car on apprend pas le même type d'information à travers les données. La méthode générative se concentre sur la structure des données de chaque classe tandis que la méthode discriminante se focalise sur la frontière de séparation entre les classes. Le tableau 3.1 nous présente quelques avantages de chaque méthode d'après le travail de Bishop et al.[76].

3.3 Description de l'apprentissage semi-supervisé soutenu

Considérons le classifieur principal par séparation $\mathcal{C}$ et le classifieur génératif $\mathcal{G}$. L'apprentissage semi-supervisé soutenu consiste à entraîner le classifieur $\mathcal{C}$ en auto-apprentissage avec l'aide du classifieur $\mathcal{G}$.

À chaque itération, le classifieur $\mathcal{G}$ est utilisé pour identifier les données sans étiquettes qui peuvent être étiquetées. La sélection est faite en utilisant le degré de confiance de classification $P(Y|X)$ disponible avec ce type de classifieur. Souvent en considérant un problème bi-classe, la fonction de classification de $\mathcal{C}$ est de la forme $Y = signe(O(X))$ où $O(X)$ représente la sortie brute du classifieur. Ainsi, les sorties brutes $O(X)$ sont calculées pour les données sans étiquettes présélectionnées par le classifieur $\mathcal{G}$. Parmi cette sélection, les données qui offrent des sorties plus élevées en valeur absolue sont ajoutées aux données étiquetées avec leur étiquettes prédites. Et le processus recommence jusqu'à épuisement de toutes les données non étiquetées. Les détails de l'apprentissage semi-supervisé soutenu sont présentés à travers l'algorithme 4.

Tableau 3.1 Modèle discriminant vs modèle génératif

Modèle génératif	**Modèle discriminant**
Ce modèle permet d'incorporer facilement les données manquantes ou partiellement étiquetées	Les classifieurs résultants ont de bonne performance de prédiction (ex. machine à vecteur de support).
Cette approche peut facilement manipuler les combinaisons (ex. reconnaissance de visage avec lunettes et/ou chapeaux, et/ou moustaches) tandis que le modèle discriminant a besoin de voir toutes les combinaisons possibles durant l'apprentissage.	Ce modèle est utilisé pour trouver la frontière précise de séparation entre les classes tandis que le modèle génératif se concentre sur la distribution des classes.
Un nouvel exemple peut être ajouté incrémentalement, parce que chaque densité de probabilité $P(X\|Y)$ est estimée séparément.	Les classifieurs discriminants sont généralement très rapide durant la phase de prédiction.

L'apprentissage semi-supervisé soutenu utilise un classifieur génératif pour aider l'apprentissage d'un classifieur discriminant. Mais, à la fin du processus, seuls les paramètres du classifieur discriminant sont retenus pour l'étape de test. Ceci différencie cette forme d'apprentissage des méthodes hybrides [53; 34; 13] parce qu'il n'y a pas de combinaison des deux classifieurs à la fin.

Algorithme 4 Apprentissage semi-supervisé soutenu (Help-Training)

Entrée L= Les exemples étiquetés, U= Les exemples non étiquetés, classifieurs $\mathcal{C}$ et $\mathcal{G}$.

Sortie Paramètres du classifieur $\mathcal{C}$

Initialiser l'ensemble actif $W = L$

tantque $U \neq \phi$ **faire**

- Entraîner les classifieurs $\mathcal{C}$ et $\mathcal{G}$ avec les données de W
- Sélectionner les exemples de U qui sont classés avec une forte probabilité avec le classifieur $\mathcal{G}$
- Calculer la sortie du classifieur $\mathcal{C}$ pour les exemples sélectionnés précédemment
- Ajouter à W les exemples de la sélection dont la sortie donne plus de confiance
- Enlever de l'ensemble U les exemples ajoutés à W.

fin tantque

3.4 Apprentissage semi-supervisé soutenu avec la SVM

Cette section décrit l'application aux SVMs de la méthode présentée dans la précédente section. Le principal classifieur (discriminant) est la SVM. En ce qui concerne le classifieur basé sur la méthode générative, nous avons choisi un classifieur utilisant une technique non-paramétrique pour estimer les densités de probabilités.

En effet, il y a deux façons d'estimer une distribution de probabilité à partir des données :

A) Méthode paramétrique. La forme fonctionnelle de la fonction de densité est spécifiée d'avance, c'est-à-dire une hypothèse est faite au préalable sur la forme de la distribution (par exemple gaussienne). Puis les paramètres de cette fonction sont estimés avec les données disponibles, en utilisant par exemple le maximum de vraisemblance.

B) Méthode non-paramétrique. Aucune hypothèse n'est faite sur la forme de la distribution. La densité de probabilité est modélisée sans faire aucune supposition sur la forme fonctionnelle de la fonction de densité recherchée.

Dans notre cas, nous avons choisi le classifieur de Parzen qui est non-paramétrique. Les estimateurs de densités sont basés sur des fonctions noyaux normalisées. En considérant un problème bi-classe, nous avons :

$$\mathcal{G}_+(x) = \frac{1}{\ell+} \sum_{i|y_i=+1} k_p(x_i, x) \tag{3.2}$$

$$\mathcal{G}_-(x) = \frac{1}{\ell-} \sum_{i|y_i=-1} k_p(x_i, x) \tag{3.3}$$

où $\ell+$ est le nombre d'exemples appartenant à la classe positive, $\ell-$ le nombre d'exemples appartenant à la classe négative et k_p est la fonction noyau de Parzen.

À chaque itération, les densités de probabilités $P(Y = -1|X)$ et $P(Y = 1|X)$ sont déterminées pour faire la présélection des données qui pourraient être étiquetées par la SVM d'après la valeur de la sortie brute qui sera calculée. Et cette estimation est mise à jour à chaque itération. L'algorithme 5 montre les différentes étapes de l'implémentation de l'apprentissage semi-supervisé soutenu avec la SVM.

3.5 Apprentissage semi-supervisé soutenu avec la LS-SVM

L'apprentissage semi-supervisé soutenu avec la LS-SVM est semblable à celui de la SVM à différence qu'avec la LS-SVM, il est possible de dériver un apprentissage incrémental.

L'apprentissage de la LS-SVM consiste à résoudre le système linéaire exprimé en (1.25). Il s'agit d'inverser une matrice afin de déterminer les paramètres $\theta = (b, \alpha)$ du classifieur. La détermination des paramètres du modèle peut alors s'écrit :

$$\begin{pmatrix} b \\ \alpha \end{pmatrix} = \begin{pmatrix} 0 & \vec{1}' \\ \vec{1} & K + \gamma^{-1} I \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ Y \end{pmatrix} \tag{3.4}$$

avec $f(x) = sign[\sum_i \alpha_i k(x_i, x) + b]$

Algorithme 5 Apprentissage semi-supervisé soutenu pour la SVM

Entrée L= Les exemples étiquetés, U= Les exemples non étiquetés.

Sortie Paramètres de la SVM

Initialiser l'ensemble actif $W = L$ et $\epsilon = \epsilon_0$

tantque $U \neq \phi$ **faire**

- Entraîner la SVM avec l'ensemble actif W
- Estimer la densité de probabilité $\mathcal{G}_+$ avec les exemples positifs de W
- Estimer la densité de probabilité $\mathcal{G}_-$ avec les exemples négatifs de W
- Sélectionner n_1 exemples de U avec une forte probabilité selon $\mathcal{G}_+$
- Sélectionner n_2 exemples de U avec une forte probabilité selon $\mathcal{G}_-$
- Calculer la sortie de la SVM pour les $(n_1 + n_2)$ exemples sélectionnés
- Former l'ensemble S à partir des (n_1+n_2) exemples sélectionnés dont la sortie $f(x) \geq \epsilon$
- Mettre à jour l'ensemble actif $W \leftarrow W \cup S$
- Mettre à jour l'ensemble des données sans étiquettes $U \leftarrow U - S$
- Réduire la valeur de ϵ si $S = \phi$

fin tantque

- Entraîner la SVM finale et retourner les paramètres

Puisqu'à chaque itération t, le classifieur est entraîné à nouveau avec juste un peu de nouvelles données qui sont étiquetées durant l'itération précédente $t-1$, nous pouvons déduire l'inverse de la nouvelle matrice augmentée en utilisant les théorèmes sur les matrices partitionnées.

Supposons à l'itération t, l'ensemble actif
$W = \{(x_1, y_1), \ldots, (x_\ell, y_\ell), (x_1^*, y_1^*), \ldots, (x_m^*, y_m^*)\}$ avec ℓ exemples étiquetées au début de l'apprentissage et m exemples sans étiquettes qui sont déjà étiquetées au cours des itérations précédentes $1, ..., t-1$, et à l'itération $t+1$, s exemples se sont ajoutés à l'ensemble actif qui devient
$W = \{(x_1, y_1), \ldots, (x_\ell, y_\ell), (x_1^*, y_1^*), \ldots, (x_m^*, y_m^*), (x_{m+1}^*, y_{m+1}^*), \ldots, (x_{m+s}^*, y_{m+s}^*)\}$.

Nous pouvons alors écrire à l'itération t,

$$\theta_t = Q_t^{-1}\widetilde{Y}_t \tag{3.5}$$

avec

$$Q_t = \begin{pmatrix} 0 & 1 & \dots & 1 & 1 & \dots & 1 \\ 1 & k(x_1,x_1)+\gamma^{-1} & \dots & k(x_1,x_\ell) & k(x_1,x_1^*) & \dots & k(x_1,x_m^*) \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & k(x_\ell,x_1) & \dots & k(x_\ell,x_\ell)+\gamma^{-1} & k(x_\ell,x_1^*) & \dots & k(x_\ell,x_m^*) \\ 1 & k(x_1^*,x_1) & \dots & k(x_1^*,x_\ell) & k(x_1^*,x_1^*)+\gamma^{-1} & \dots & k(x_1^*,x_m^*) \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & k(x_m^*,x_1) & \dots & k(x_m^*,x_\ell) & k(x_m^*,x_1^*) & \dots & k(x_m^*,x_m^*)+\gamma^{-1} \end{pmatrix} \tag{3.6}$$

et

$$\widetilde{Y}_t = (0, y_1, \dots, y_\ell, y_1^*, \dots, y_m^*)' . \tag{3.7}$$

Et à l'itération $t+1$,

$$\theta_{t+1} = Q_{t+1}^{-1}\widetilde{Y}_{t+1} \tag{3.8}$$

avec

$Q_{t+1} =$

$$\begin{pmatrix} 0 & 1 & \dots & 1 & 1 & \dots & 1 \\ 1 & k(x_1,x_1)+\gamma^{-1} & \dots & k(x_1,x_\ell) & k(x_1,x_1^*) & \dots & k(x_1,x_{m+s}^*) \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & k(x_\ell,x_1) & \dots & k(x_\ell,x_\ell)+\gamma^{-1} & k(x_\ell,x_1^*) & \dots & k(x_\ell,x_{m+s}^*) \\ 1 & k(x_1^*,x_1) & \dots & k(x_1^*,x_\ell) & k(x_1^*,x_1^*)+\gamma^{-1} & \dots & k(x_1^*,x_{m+s}^*) \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & k(x_{m+s}^*,x_1) & \dots & k(x_{m+s}^*,x_\ell) & k(x_{m+s}^*,x_1^*) & \dots & k(x_{m+s}^*,x_{m+s}^*)+\gamma^{-1} \end{pmatrix} \tag{3.9}$$

et

$$\widetilde{Y}_{t+1} = \left(0, y_1, \dots, y_\ell, y_1^*, \dots, y_{m+s}^*\right)'. \tag{3.10}$$

En partitionnant la matrice Q_{t+1} comme suit :

$$Q_{t+1} = \begin{pmatrix} Q_t & K_{s1} \\ K_{s1}' & K_{ss} \end{pmatrix} \tag{3.11}$$

avec

$$K_{s1} = \begin{pmatrix} 1 & \dots & 1 \\ k(x_1,x_{m+1}^*) & \dots & k(x_1,x_{m+s}^*) \\ \dots & \dots & \dots \\ k(x_\ell,x_{m+1}^*) & \dots & k(x_\ell,x_{m+s}^*) \\ k(x_1^*,x_{m+1}^*) & \dots & k(x_1^*,x_{m+s}^*) \\ \dots & \dots & \dots \\ k(x_m^*,x_{m+1}^*) & \dots & k(x_m^*,x_{m+s}^*) \end{pmatrix} \tag{3.12}$$

et

$$K_{ss} = \begin{pmatrix} k(x^*_{m+1}, x^*_{m+1}) + \gamma^{-1} & \dots & k(x^*_{m+1}, x^*_{m+s}) \\ \dots & \dots & \dots \\ k(x^*_{m+s}, x^*_{m+1}) & \dots & k(x^*_{m+s}, x^*_{m+s}) + \gamma^{-1} \end{pmatrix} \tag{3.13}$$

Nous pouvons déduire son inverse avec le théorème sur l'inverse des matrices partitionnées,

$$Q^{-1}_{t+1} = \begin{pmatrix} Q_t^{-1} + Q_t^{-1} K_{s1} D'_s & -D_s \\ -D'_s & C_s \end{pmatrix} \tag{3.14}$$

avec

$$C_s = \left[K_{ss} - K'_{s1} Q_t^{-1} K_{s1}\right]^{-1} \tag{3.15}$$

$$D_s = Q_t^{-1} K_{s1} C_s \tag{3.16}$$

Ainsi, avec la formule (3.14), nous pouvons faire la mise à jour des paramètres du classifieur à chaque itération avec moins de temps de calcul, puisque la complexité de calcul de l'inverse de Q_{t+1} est réduire de $\mathcal{O}((\ell + m + 1)^3 + s(\ell + m + 1)^2 + s^2(\ell + m + 1))$ opérations [1].

1. L'évaluation a été faite en considérant que le produit entre une matrice de taille $m \times p$ et une autre de taille $p \times n$ nécessite $\mathcal{O}(mnp)$ opérations tandis que l'inverse d'une matrice carrée requiert $\mathcal{O}(n^3)$ opérations.

CHAPITRE 4

SÉLECTION DE MODÈLE

4.1 Introduction

La sélection de modèle est le processus qui consiste à choisir les hyperparamètres en vue d'avoir la meilleure performance du classifieur. Les hyperparamètres sont des paramètres fixés d'avance avant de procéder à l'entraînement du classifieur et qui ont un impact sur ce dernier. Les SVMs et les LS-SVMs ont deux types d'hyperparamètres : les paramètres de régularisation γ ou C et les paramètres du noyau. Dans ce chapitre, nous allons aborder la sélection de modèle pour les SVMs et les LS-SVMs en mode supervisé et aussi en mode semi-supervisé.

4.2 Optimisation des hyperparamètres

Technique 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 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 des données disponibles 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.

Erreur empirique

En pratique, l'optimisation des hyperparamètres se fait en minimisant une fonction décrivant la capacité de généralisation du classifieur. Il s'agit du *critère de sélection de modèle* évalué sur des données qui n'ont pas participé à l'apprentissage afin de minimiser le biais. Plusieurs types de critères ont été proposés pour faire la sélection de modèle pour les SVMs ; parmi ces critères, nous avons retenu l'erreur empirique qui est une fonction linéaire simple ne nécessitant pas la résolution d'un problème d'optimisation quadratique excepté celui de l'apprentissage de la SVM.

La sélection de modèle pour les SVMs en utilisant le critère de l'erreur empirique a été développée par Ayat au cours de sa 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, l'hyperplan optimal de séparation entre les deux classes de données est défini par :

$$f(x) = sign[\sum_{j=1}^{N_{VS}} \alpha_j y_j k(x_j, x) + b] \tag{4.1}$$

où N_{VS} désigne le nombre de vecteurs de support.

En posant $t_i = (y_i + 1)/2$, l'étiquette bipolaire des observations devient uni-polaire ; 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 :

$$E_i = |t_i - \hat{p}_i| \tag{4.2}$$

où $\hat{p}_i$ représente une estimation 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'une SVM est réalisée en utilisant la fonction logistique proposée par Platt dans [62]. Cette fonction possède deux paramètres A et B et s'écrit :

$$\hat{p}_i = \frac{1}{1 + exp[A.f(x_i) + B]} \tag{4.3}$$

4.3 Sélection de modèle pour les SVMs supervisées

La sélection de modèle pour les SVMs supervisées se fait en utilisant un ensemble de validation $\mathcal{D}_v$ étiqueté autre que l'ensemble d'apprentissage. En supposant que la fonction noyau dépend d'un ou plusieurs paramètres que nous notons sous forme de vecteur $\lambda = (\lambda_1, \lambda_2, ...)$ en incluant l'hyperparamètre de régularisation, les valeurs optimales des hyperparamètres sont obtenues en minimisant la somme de l'erreur empirique évaluée sur $\mathcal{D}_v$.

$$\lambda_{optimal} = \underset{\lambda}{\operatorname{argmin}} \sum_{(x_i, y_i) \in \mathcal{D}_v} E_i \tag{4.4}$$

La minimisation de ce critère peut se fait de façon automatique en utilisant un algorithme de descente de gradient comme décrit dans l'agorithme 6, ou en se servant d'un programme de minimisation numérique.

4.4 Sélection de modèle pour les LS-SVMs supervisées

La validation croisée avec $k = \ell$ taille de l'ensemble de validation s'appelle "leave-one-out" (LOO). La procédure LOO est très coûteux en temps de calcul parce que cela nécessite ℓ fois l'apprentissage du classifieur en omettant chaque fois l'élément utilisé pour estimer la performance. Ainsi, avec les SVMs, cette méthode n'est presque pas utilisée ; on utilise plutôt des estimations de LOO ou des bornes de LOO [84; 83; 43; 46; 60; 59; 23; 79; 36].

Algorithme 6 Sélection de modèle de la SVM avec l'erreur empirique

Entrées : Ensemble d'apprentissage $\mathcal{D}$, ensemble de validation $\mathcal{D}_v$, type de noyau, pas du gradient η.

Sortie : Hyperparamètres optimaux λ.

Initialiser les hyperparamètres

tantque $\left|\frac{\partial E}{\partial \lambda}\right| \geq \epsilon$ **faire**

- Entraîner la SVM sur l'ensemble $\mathcal{D}$ avec les hyperparamètres courants
- Estimer les probabilités $\hat{p}_i$
- Calculer le gradient de l'erreur $\left|\frac{\partial E}{\partial \lambda}\right|$
- Corriger les hyperparamètres selon le gradient $\lambda^{t+1} = \lambda^t - \eta \frac{\partial E}{\partial \lambda}$

fin tantque

Cependant, la formulation des LS-SVMs permet d'utiliser la procédure LOO avec un seul apprentissage [17]. La prédiction LOO pour l'exemple x_i s'écrit :

$$f^{(-i)}(x_i) = y_i - \frac{\alpha_i}{H_{ii}^{-1}} \tag{4.5}$$

avec

$$H = \begin{pmatrix} K + \gamma^{-1} I & \vec{1}' \\ \vec{1} & 0 \end{pmatrix} \tag{4.6}$$

Partant de ce résultat, nous pouvons estimer l'erreur empirique à travers une procédure LOO avec les LS-SVMs avec moindre coût. Dans ce cas, un ensemble de validation distinct n'est pas nécessaire. Tout l'ensemble d'apprentissage $\mathcal{D}$ est utilisé.

$$\lambda_{optimal} = \underset{\lambda}{\operatorname{argmin}} \sum_{(x_i, y_i) \in \mathcal{D}} \left| \frac{y_i + 1}{2} - \frac{1}{1 + exp[A.f^{(-i)}(x_i) + B]} \right| \tag{4.7}$$

4.5 Sélection de modèle pour les SVMs/LS-SVMs semi-supervisées

La difficulté de l'apprentissage semi-supervisé consiste en ce que nous ne disposons que de peu de données avec étiquette. Ainsi, il est incommode de réduire cet ensemble de données étiquetées pour former un ensemble de validation. Sinon, ce serait une application directe de la sélection de modèle en apprentissage supervisé.

Nous proposons alors une procédure de validation croisée uniquement sur les données avec étiquettes. Il s'agit donc de subdiviser les données étiquetées en k sous-ensembles et d'utiliser à chaque itération les $k-1$ sous-ensembles avec les données non étiquetées pour l'apprentissage et d'estimer le critère de sélection de modèle sur le sous-ensemble omis.

$$\lambda_{optimal} = \underset{\lambda}{\operatorname{argmin}} \sum_{t=1}^{k} \sum_{(x_i,y_i)\in\mathcal{V}_t} \left| \frac{y_i+1}{2} - \frac{1}{1+exp[A.f^{(-\mathcal{V}_t)}(x_i)+B]} \right| \tag{4.8}$$

où :

$\mathcal{V}_t$ est le sous-ensemble des données étiquetées omis,

$\mathcal{V}_1 \cup \mathcal{V}_2 \ldots \mathcal{V}_k = \mathcal{D}_l$ désigne l'ensemble des données d'apprentissage étiquetées,

k est le nombre de validations croisées,

$f^{(-\mathcal{V}_t)}$ est le classifieur entrainé sans le sous-ensemble $\mathcal{V}_t$, mais avec les données sans étiquettes.

En ce qui concerne les LS-SVMs, nous allons utiliser la procédure LOO évaluée uniquement sur les données étiquetées. Il suffit donc de sommer l'erreur empirique des ℓ premiers éléments et de laisser les n éléments restants de la base d'apprentissage, puisque nous ne connaissons pas leur réelle étiquette.

CHAPITRE 5

EXPÉRIMENTATION, RÉSULTATS ET DISCUSSION

5.1 Introduction

Le présent chapitre est consacré aux diverses expérimentations, ainsi que les résultats et conclusions qui en découlent. Cependant, nous invitons le lecteur à consulter les articles en annexes pour plus de détails. Nous avons testé les différents algorithmes proposés pour la tâche d'apprentissage semi-supervisé sur des données artificielles et réelles. Dans un premier temps, nous allons montré l'avantage d'utiliser les données non étiquetées pour améliorer les frontières de décisions des classifieurs. Ainsi, nous allons procéder à une comparaison avec l'apprentissage supervisé. Dans un second temps, nous allons comparer nos résultats avec d'autres obtenus avec des algorithmes d'apprentissage semi-supervisé. Enfin, nous allons étudier certaines propriétés de nos algorithmes.

5.2 Bases de données

5.2.1 Gaus50

Ce sont des données artificielles de dimension 50 générées à partir de deux gaussiennes ayant même matrice de covariance qui est la matrice identité. Nous avons un problème à deux classes équiprobables [37]. La moyenne de la gaussienne de la classe 1 est $(a, a, ..., a)$ et celle de la classe 2 est $-(a, a, ..., a)$. Dans le but d'obtenir l'erreur de Bayes égal à 5%, le paramètre a est fixé à 0.23. Nous avons produire 550 exemples pour l'apprentissage et 1000 exemples pour le test.

5.2.2 Gaus50x

Nous avons aussi produit une autre base de données artificielles de dimension 50. Les données des deux classes sont générées de façon équiprobable à partir de mélange gaussien avec un recouvrement significatif entre les classes dont les probabilités conditionnelles s'écrivent : $p(x|y=1) = 0.49\mathcal{N}(\mu_1, I) + 0.51\mathcal{N}(\mu_2, I)$ et $p(x|y=-1) = 0.49\mathcal{N}(-\mu_1, I) + 0.51\mathcal{N}(-\mu_2, I)$ avec $\mu_1 = (0.25, 0.25, ..., 0.25, 0.25)$, $\mu_2 = (0.25, 0.25, ..., -0.25, -0.25)$, $N(\mu, I)$ représente une distribution gaussienne de vecteur moyenne μ et de matrice covariance identité I. Nous avons aussi généré 550 exemples pour l'apprentissage et 1000 exemples pour le test.

5.2.3 Text

Cette base de données décrit un réel problème de classification de texte. Nous avons utilisé les classes *Mac* et *Windows* prises d'un ensemble de données constitué à partir d'un forum de discussion. Ce sont des données de grande dimension, soit 7511 et elles sont pré-traitées comme dans [74]. Nous avons divisé la base en deux sous-ensembles : 1446 exmples pour l'apprentissage et 500 pour le test.

5.2.4 USPS

USPS bien connu sous le nom "US Postal Service" décrit un problème de classification de chiffres manuscrits. Les chiffres sont représentés par des images normalisées en niveau de gris de taille 16×16. L'ensemble d'apprentissage contient 7291 exemples tandis que celui de test est formé de 2007 exemples. Nous avons un problème multiclasse avec 10 classes (les chiffres 0 à 9). Pour la résolution, le problème est décomposé avec la stratégie *un-contre-tous*.

5.3 Comparaison avec l'apprentissage supervisé

Pour réaliser cette comparaison, nous avons effectué divers tests en utilisant respectivement les SVMs et les LS-SVMs :

1) Comparaison avec SVM

En premier lieu, une SVM supervisée a été entraînée avec toute l'ensemble d'apprentissage avec les étiquettes réelles. Ce qui constitue la référence. Ensuite, une portion de 10% des données d'apprentissage a été sélectionnée pour former l'ensemble des données étiquetées et le reste est considéré comme sans étiquette. L'ensemble des données étiquetées a servi pour entraîner une autre SVM en mode supervisé puis deux SVMs semi-supervisé (S^3VM-GA et $HT-S^3VM$) avec les données sans étiquettes. Ainsi, nous avons construit trois classifieurs qui sont testés sur la base de test et comparés avec la référence.

2) Comparaison avec LS-SVM

Nous avons répéter la même procédure en utilisant la LS-SVM, mais avec une portion de 5% [1] des données d'apprentissage sélectionnées pour former l'ensemble des données étiquetées. En plus, nous avons construire quatre classifieurs LS-SVM semi-supervisés ($S^2LS-SVM_1$, $S^2LS-SVM_2$, $HTS^2LS-SVM$ et $BS^2LS-SVM$).

La sélection des données étiquetées a été répétée 10 fois. Les figures 5.1, 5.2 et 5.3 montrent un sommaire des résultats obtenus. Nous constatons que l'utilisation des données sans étiquette améliore de façon significative la performance du classifieur résultant. De plus, la performance de généralisation du classifieur en semi-supervisé tend à atteindre celle de la référence, c'est-à-dire celle du classifieur qu'on obtiendrait avec toutes les données étiquetées.

5.4 Comparaison avec d'autres techniques d'apprentissage semi-supervisé

Dans cette section, nous allons procéder à la comparaison des divers algorithmes d'apprentissage proposés dans cette thèse avec d'autres de la même famille. Toutefois, le domaine de semi-supervisé étant moins défriché en comparaison au supervisé, nous n'avons pas trouvé d'algorithmes disponibles pour la LS-SVM semi-supervisée. Ainsi, en plus de nos algorithmes,

1. Nous avons remarqué que la LS-SVM semi-supervisé est robuste avec cette portion de données étiquetées et donne des résultats intéressants, ce qui n'est pas le cas de la SVM semi-supervisé.

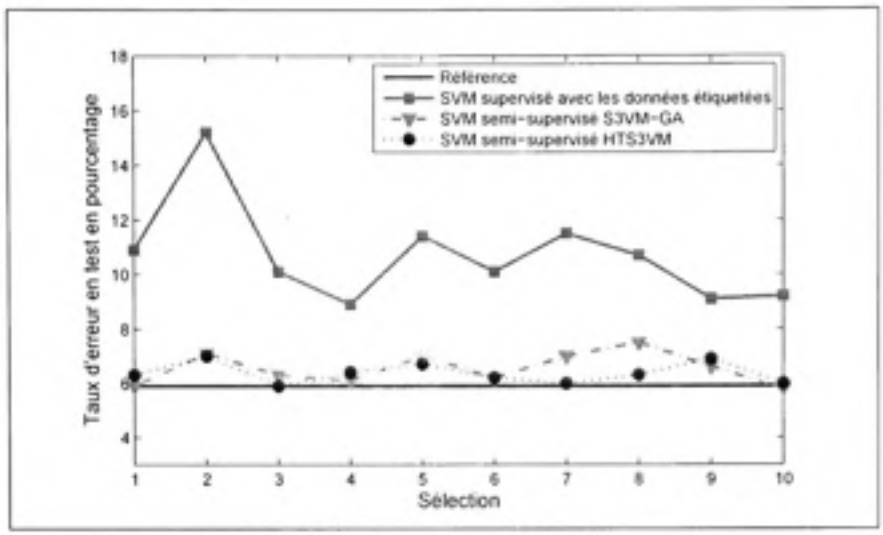


Figure 5.1 Comparaison entre le supervisé et le semi-supervisé avec la SVM sur la base Gaus50.

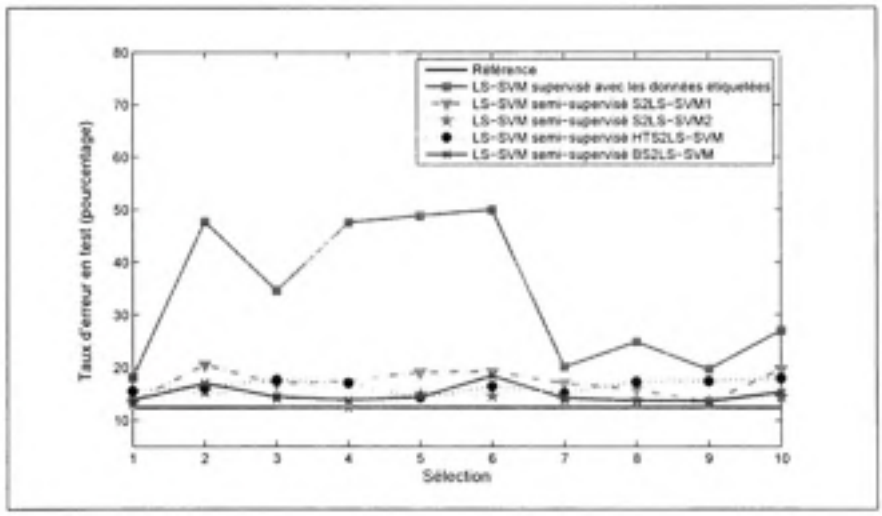


Figure 5.2 Comparaison entre le supervisé et le semi-supervisé avec la LS-SVM sur la base Gaus50x.

nous avons testé deux autres algorithmes de S^3VM dont les codes sont disponibles sur le web. Il s'agit de la $S^3VM - light$ [48] qui est la première implémentation de la S^3VM et la $S^3VM - CCCP$ [28] basée sur un processus de décomposition de la fonction objective.

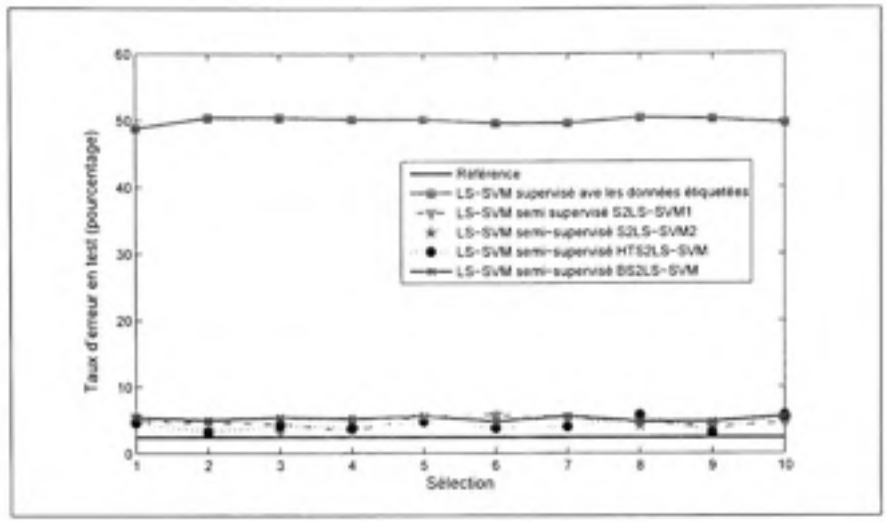


Figure 5.3 Comparaison entre le supervisé et le semi-supervisé avec la LS-SVM sur la base Text.

Les résultats sont résumés dans le tableau 5.1. Une première analyse des résultats montre que la performance des différents algorithmes sont approximativement identiques. Cependant, une observation approfondie indique que :

- aucun algorithme ne surpasse les autres en performance sur toutes les bases,

- certains de nos algorithmes sont plus performants que la $S^3VM-light$ et la $S^3VM-CCCP$ sur toutes les bases testées.

5.5 Impact du nombre d'exemples étiquetés

Nous avons testé dans cette section le comportement de chacun de nos algorithmes en fonction du ratio N_l/N_u, où N_l désigne le nombre d'exemples sans étiquettes et N_u le nombre d'exemples avec étiquettes. Les expérimentations ont été effectuées sur les bases Gaus50 et Text. Les résultats sont montrés en figures 5.4, 5.5, 5.6 et 5.7.

Tableau 5.1 Comparaison entre les algorithmes d'apprentissage semi-supervisé proposés et d'autres SVMs semi-supervisées, $S^3VM - light$ [48], $S^3VM - CCCP$ [28], dont les codes sont disponibles sur le web

	Gaus50	**Gaus50x**	**Texte**	**USPS**
$S^3VM - light$	8.80%	16.30%	5.40%	14.70%
$S^3VM - CCCP$	7.70%	15.50%	5.45%	7.32%
$S^3VM - GA$	5.70%	14.70%	**4.00%**	7.62 %
HTS^3VM	6.30%	17.50%	*4.20%*	7.77%
$S^2LS - SVM_1$	6.10%	16.50%	5.00%	10.31%
$S^2LS - SVM_2$	*5.50%*	*14.00%*	*4.20%*	*6.72 %*
$BS^2LS - SVM$	**5.20%**	**12.00%**	4.80%	6.92%
$HTS^2LS - SVM$	5.90%	15.80%	*4.20%*	**6.02%**

Les résultats concernant S^3VM montrent que la $S^3VM - GA$ a une performance supérieure au HTS^3VM. Quant au $S^2LS - SVM$, nous remarquons que $S^2LS - SVM_2$ et $BS^2LS - SVM$ sont plus robustes lorsque nous avons une très petite quantité de données étiquetées ; tandis que les performances de tous les algorithmes se rejoignent au fur et à mesure que le ratio N_l/N_u devient important.

5.6 Apport de la sélection de modèle

La sélection de modèle est un processus très important dans la construction d'un classifieur. Et nous pouvons voir l'impact de cette étape sur la performance des LS-SVMs en mode supervisé dans le dernier article présenté en annexe. Dans cette section, nous allons montré l'intérêt de la stratégie proposée pour la sélection de modèle dans le contexte de l'apprentissage semi-supervisé.

Dans un premier temps, nous considérons le mode supervisé avec les étiquettes réelles. En utilisant la méthode de validation croisée, les hyperparamètres du classifieurs sont optimisés.

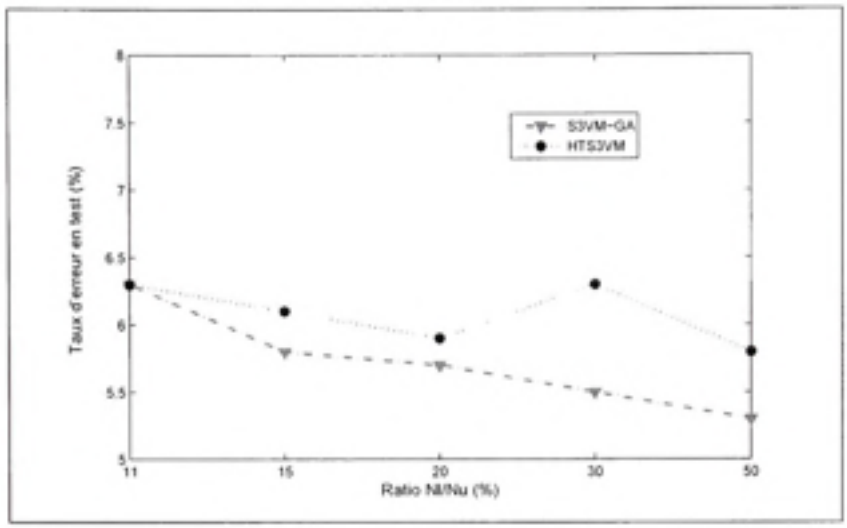


Figure 5.4 Variation de la performance des classifieurs S^3VM en fonction du ratio $\frac{N_l}{N_u}$ sur la base Gaus50.

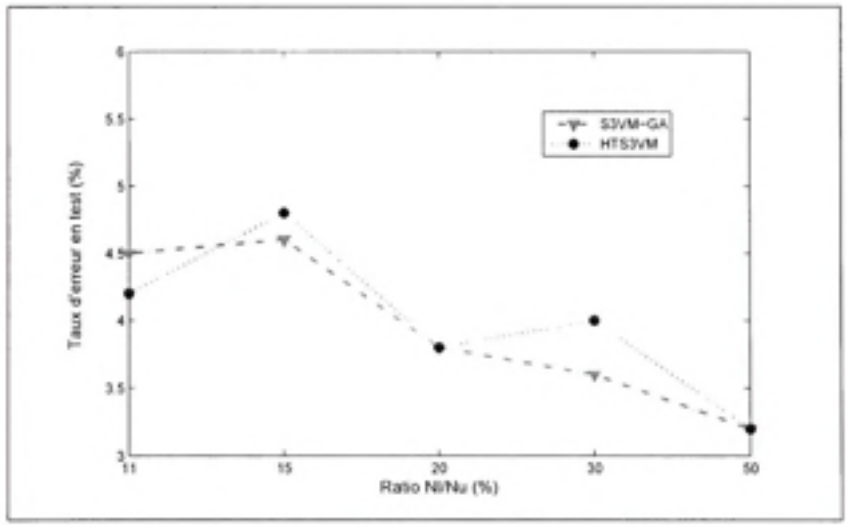


Figure 5.5 Variation de la performance des classifieurs S^3VM en fonction du ratio $\frac{N_l}{N_u}$ sur la base Text.

Puis, nous nous servons de ces valeurs comme valeurs optimales pour l'apprentissage semi-supervisé. Les machines ainsi construits constituent les références. Dans la vraie vie, nous ne

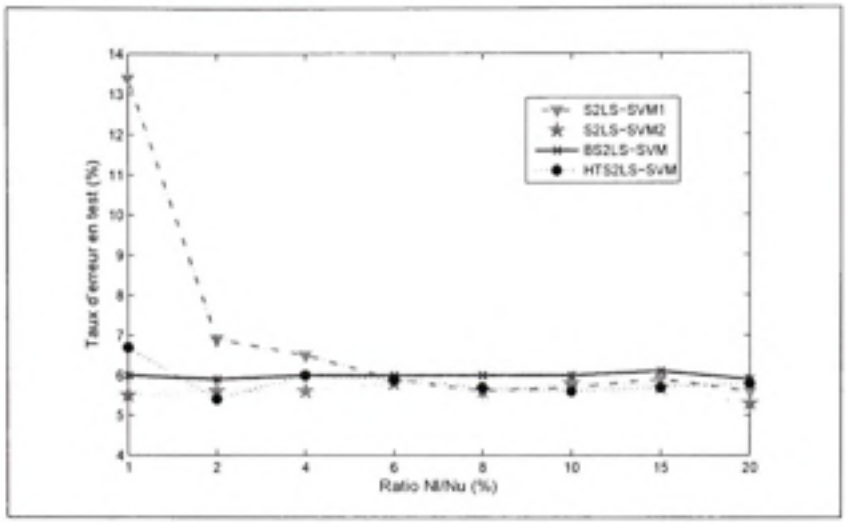


Figure 5.6 Variation de la performance des classifieurs $S^2LS-SVM$ en fonction du ratio $\frac{N_l}{N_u}$ sur la base Gaus50.

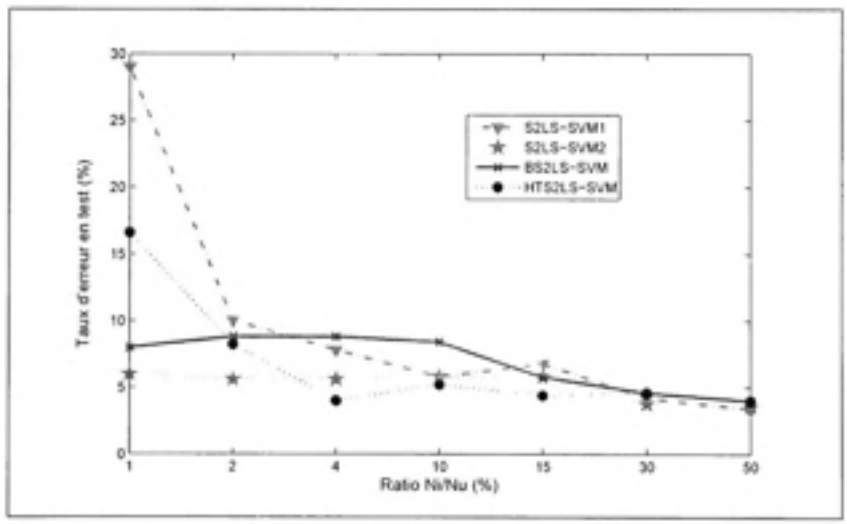


Figure 5.7 Variation de la performance des classifieurs $S^2LS-SVM$ en fonction du ratio $\frac{N_l}{N_u}$ sur la base Text.

connaissons pas les étiquettes réelles de toutes les données, donc les classifieurs de référence sont construits seulement à des fins de comparaison.

Dans un second temps, nous utilisons les stratégies décrites dans le chapitre 4 pour entraîner d'autres classifeurs en mode semi-supervisé en incluant la sélection de modèle. Nous avons entraîné la SVM et la $LS-SVM$ semi-supervisé avec la méthode soutenu (Help-Training). Les résultats obtenus sont résumés dans les tableaux 5.2 et 5.3, où nous pouvons noter que la stratégie proposée pour faire la sélection de modèle en semi-supervisé donne des solutions satisfaisantes. Cependant, nous pouvons remarquer que parfois l'algorithme de sélection de modèle peut converger dans de minimum local, ce qui dégrade la performance du classifieur.

Tableau 5.2 Mise en évidence de notre technique de sélection de modèle en semi-supervisé avec la base "Gaus50"

Sélection	**SVM semi-supervisé**		**LS-SVM semi-supervisé**	
	Référence	Notre Méthode	Référence	Notre Méthode
#1	6.3	6.5	5.3	5.6
#2	7.0	7.3	5.8	5.6
#3	5.9	6.4	5.9	6.2
#4	6.4	6.6	6.2	5.7
#5	6.7	6.4	6.2	5.9
#6	6.2	6.7	6.1	6.4
#7	6.0	6.2	6.4	6.3
#8	6.3	6.7	5.5	6.0
#9	6.9	8.3	5.6	5.7
#10	6.0	6.2	5.7	5.9
Moyenne	6.37	6.73	5.87	5.93

5.7 Conclusion

Dans ce chapitre, nous avons présenté quelques résultats commentés des multiples expérimentations faites avec les algorithmes que nous avons développés au cours de nos recherches doctorales. Nous invitons le lecteur à consulter les articles de journaux en annexes pour consulter

Tableau 5.3 Mise en évidence de notre technique de sélection de modèle en semi-supervisé avec la base "Text"

Sélection	**SVM semi-supervisé**		**LS-SVM semi-supervisé**	
	Référence	Notre Méthode	Référence	Notre Méthode
#1	3.8	4.6	4.8	4.2
#2	3.8	3.8	4.6	4.0
#3	4.2	4.2	4.8	3.6
#4	3.2	3.0	4.8	2.8
#5	4.2	4.6	5.0	4.6
#6	4.0	4.0	3.4	4.2
#7	4.0	4.0	4.2	3.8
#8	2.6	3.4	5.8	3.4
#9	3.6	2.8	5.0	3.0
#10	4.2	4.6	6.2	4.4
Moyenne	3.76	3.90	4.86	3.80

tous les détails entourant les expérimentations. Aussi, des expérimentations complémentaires sont disponibles dans les divers articles.

En somme, nous pouvons noter que la performance des algorithmes sont similaires lorsque la quantité des données étiquetées est importante. Mais, lorsque peu de données étiquetées sont disponibles, deux algorithmes $S^2LS - SVM_2$ et $BS^2LS - SVM$ se distinguent des autres par leur robustesse.

CONCLUSION

Les SVMs et les LS-SVMs sont des classifieurs qui ont eu beaucoup de succès en mode supervisé. Leur pouvoir de généralisation n'est plus à discuter et leur application dans plusieurs problèmes de reconnaissance de formes demeure remarquable. Cependant, l'entraînement de ces machines avec des données partiellement étiquetées reste un problème très intéressant. Car, l'apprentissage semi-supervisé permet non seulement de réduire le coût d'étiquetage des données en ne marquant qu'une portion des données, mais aussi il favorise le développement des systèmes évolutifs et adaptatifs, c'est-à-dire qui s'améliorent au cours de la phase de test. Ainsi, dans cette thèse, nous nous sommes penchés sur la question d'apprentissage semi-supervisé où les données sans et avec étiquettes sont utilisées pour entraîner les classifieurs SVMs et LS-SVMs.

Nous avons proposé plusieurs algorithmes avec des caractéristiques variables. Pour les SVMs, nous avons développé deux sortes d'algorithmes $S^3VM - GA$ et $HT - S^3VM$ tandis que pour les LS-SVMs, nous avons proposé quatre techniques qui sont dénommés $S^2LS - SVM_1$, $S^2LS - SVM_2$, $HTS^2LS - SVM$ et $BS^2LS - SVM$.

En général, les divers algorithmes proposés sont basés sur deux principes essentiels : l'inférence bayésienne et l'apprentissage soutenu. Ces deux idées maîtresses ont constitué les piliers de nos travaux de recherche doctorale en vue de concevoir des algorithmes d'apprentissage semi-supervisé.

En se basant sur la théorie bayésienne, nous avons énoncé une formulation pour l'apprentissage semi-supervisé avec inférence bayésienne à un et deux niveaux. Avec l'inférence à un niveau, les étiquettes non connues et le paramètre du classifieur sont estimés en une seule étape à partir des distributions *a priori* et *a posteriori* du modèle choisi. Alors que l'inférence à deux niveaux permet de découpler l'inférence des étiquettes non connues et du paramètre en deux phases à l'aide d'une méthode hiérarchique. L'application de la stratégie d'inférence à un niveau a permis de formuler et d'implémenter les algorithmes $S^3VM - GA$, $S^2LS - SVM_1$

et $S^2LS - SVM_2$; cependant, à partir de l'inférence à deux niveaux, nous avons déduit l'algorithme $BS^2LS - SVM$.

Le deuxième principe développé au cours de cette thèse est l'apprentissage semi-supervisé soutenu (Help-Training) où le principal classifieur qui est discriminant est aidé par un classifieur d'approche générative au cours d'un processus d'auto-apprentissage. Cette technique a permis d'intégrer les avantages d'un modèle génératif dans le mécanisme d'apprentissage d'un classifieur discriminant. La stratégie a été appliquée avec succès aux SVMs et LSVMs avec le classifier génératif de Parzen. Les algorithmes qui en découlent sont $HT - S^3VM$ et $HTS^2LS - SVM$.

Pour valider nos diverses approches, nous avons testé les algorithmes développés sur des données artificielles et sur des problèmes réelles : classification de textes et reconnaissance de chiffres manuscrits à l'aide des données provenant de USPS. Les résultats expérimentaux obtenus ont démontré l'intérêt des algorithmes conçus. Nous avons aussi remarqué que deux de ces algorithmes, $S^2LS - SVM_2$ et $BS^2LS - SVM$, se sont montrés plus robustes que les autres lorsqu'il n'y a que très peu de données étiquetées et dans le cas où la proportion des données avec étiquettes est significative, tous les algorithmes développés s'équivalent approximativement en performance. Le temps computationnel de chaque algorithme est donné dans les articles correspondants qui sont annexés à ce document, nous pouvons remarquer qu'excepté le $S^3VM - GA$, les algorithmes $S^2LS - SVM_2$ et $BS^2LS - SVM$ sont les plus gourmands en temps de calcul.

Perspectives

Parmi tous les algorithmes développés, le temps computationnel du $S^3VM - GA$ s'est révélé très important. Un travail sur ce point est nécessaire afin de réduire ce temps. Nous pensons qu'en injectant plus de connaissance a priori sur le modèle d'optimisation, on pourrait arriver à réduire l'espace de recherche et par conséquent réduire le temps computationnel.

L'approche bayésienne est basée sur la distribution *a priori*. Ainsi, changer la distribution *a priori* signifie changer les résultats. Par conséquent, l'amélioration des techniques proposées peut être réalisée en faisant une étude rigoureuse sur la définition des distributions *a priori* utilisées. Une connaissance plus approfondie du problème étudié peut amener à mieux spécifier les distributions a priori afin de perfectionner le modèle proposé.

Le principal but visé par l'apprentissage semi-supervisé est d'égaler la performance du classifieur qu'on aurait obtenu en utilisant toutes les données avec leurs réelles étiquettes. Mais, nous avons montré par des expérimentations dans le chapitre 5 que cette performance dépend du nombre de données étiquetées disponibles jusqu'à un certain seuil. Alors, la question essentielle qui provient de la précédente remarque est comment procéder à la détermination automatique du pourcentage de données à étiqueter afin d'avoir une performance acceptable. Nous sommes convaincus que cette valeur dépendra des spécificités des données décrivant le problème traité. Les spécificités qui rentrent en compte pourraient être le nombre de caractéristiques, le chevauchement entre les classes, le nombre de classes, l'équilibre des données entre les classes, etc.

Enfin, les deux grandes idées autours desquelles nous avons bâti et développé les divers algorithmes d'apprentissage semi-supervisé peuvent être appliquées à d'autres types de classifieurs. Ces concepts étant élaborés indépendamment des classifieurs constituent de précieuse références pour développer d'autres techniques. Ainsi, l'extension des méthodes proposées peuvent conduire à la conception de nouveaux algorithmes d'apprentissage semi-supervisé.

BIBLIOGRAPHIE

[1] A.Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proc. 18th International Conf. on Machine Learning*, pages 19–26. Morgan Kaufmann, San Francisco, CA, 2001.

[2] Mathias M. Adankon and Mohamed Cheriet. New formulation of SVM for model selection. In IEEE, editor, *International Joint Conference in Neural Networks 2006*, pages 3566–3573, Vancouver, BC, 2006.

[3] Mathias M. Adankon and Mohamed Cheriet. Optimizing resources in model selection for support vector machines. *Pattern Recognition*, 40(3) :953–963, 2007.

[4] Mathias M. Adankon and Mohamed Cheriet. Help-training for semi-supervised discrimininative classifier, application to SVM. In *19th Internatioonale Conference on Pattern Recognition*, Tampa, USA, 2008.

[5] Mathias M. Adankon, Mohamed Cheriet, and Nedjem E. Ayat. Optimizing resources in model selection for support vector machines. In IEEE, editor, *International Joint Conference in Neural Networks*, volume 2, pages 925–930, Montreal, 2005.

[6] N. E. Ayat. *Sélection automatique de modèle des machines à vecteurs de support : Application à la reconnaissance d'images de chiffres manuscrits*. PhD thesis, École de Technologie Supérieure, University of Quebec, 2003.

[7] N. E. Ayat, M. Cheriet, and C.Y. Suen. Automatic model selection for the optimization of the SVM kernels. *Pattern Recognition*, 38(10) :1733–1745, 2005.

[8] Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and semi-supervised learning on large graphs. *Lecture Notes in Computer Science*, 3120 :624–638, January 2004.

[9] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization : A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7 :2399–2434, 2006.

[10] K. Bennett and A. Demiriz. Semi-supervised support vector machines. *Advances in Neural Information Processing Systems*, 11 :368–374, 1998.

[11] T. De Bie and N. Cristianini. Convex methods for transduction. In S. Thrun, L. Saul, and B. SchÄolkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2004.

[12] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann*

Publishers, 1998.

[13] A. Bosch, A. Zisserman, and X. Muoz. Scene classification using a hybrid generative/discriminative approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(4) :712–727, 2008.

[14] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Computational Learing Theory*, pages 144–152, 1992.

[15] E. J. Bredensteiner and K. Bennett. Multicategory classification by support vector machines. *Computational Optimization and Applications*, 12 :53–79, 1999.

[16] V. Castelli and T. Cover. The relative value of labeled and unlabeled samples in pattern recognition with unknown mixing parameter. *IEEE Transactions on Information Theory*, 42 :2101–2117, 1996.

[17] Gavin Cawley. Leave-one-out cross-validation based model selection criteria for weighted LS-SVMs. In *proceedings IJCNN 2006, Vancouver, Canada*, July 2006.

[18] Gavin C. Cawley and Nicola L. C. Talbot. Improved sparse least-squares support vector machines. *Neurocomputing*, 48 :1025–1031, 2002.

[19] Gavin C. Cawley and Nicola L. C. Talbot. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural Networks*, 17 :1467–1475, 2004.

[20] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[21] O. Chapelle, V. Sindhwani, and S. S. Keerthi. Branch and bound for semi-supervised support vector machines. In *Proceedings of the NIPS 2006*, 2006.

[22] O. Chapelle, V. Sindhwani, and S.S. Keerthi. Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research*, 9 :203–233, 2008.

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

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

[25] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

[26] W. Chu, C. J. Ong, and S. S. Keerthi. An improved conjugate gradient scheme to the solution of least squares SVM. *IEEE Transactions on Neural Networks*, 16(2) :498–501, 2005.

[27] Kok Seng Chua. Efficient computations for large least square support vector machine classifiers. *Pattern Recognition Letters*, 24 :75–80, 2003.

[28] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Large scale transductive SVM. *Journal of Machine Learning Research*, 7 :1687–1712, 2006.

[29] Fabio Gagliardi Cozman, Ira Cohen, and Marcelo Cesar Cirelo. Semi-supervised learning of mixture models. In *ICML*, pages 99–106, 2003.

[30] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[31] B. J. de Kruif and T. J. deVries. Pruning error minimization in least squares support vector machines. *IEEE Transactions on Neural Networks*, 14 :696–702, 2003.

[32] A. Demirez and K. Bennett. Optimization approaches to semi-supervised learning. In *Applications and Algorithms of Complementarity, M. Ferris, O. Mangasarian, and J. Pang, editors. Kluwer Academic Publishers, Boston*, 2000.

[33] A. Demiriz, K. Bennett, and M. Embrechts. Semi-supervised clustering using genetic algorithms. Technical report, R.P.I. Math Report No. 9901, Rensselaer Polytechnic Institute, Troy, New York, 1999.

[34] Gregory Druck, Chris Pal, Andrew McCallum, and Xiaojin Zhu. Semi-supervised classification with hybrid generative/discriminative methods. In *KDD '07 : Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 280–289, New York, NY, USA, 2007. ACM.

[35] G. Fung and O. Mangasarian. Semi-supervised support vector machines for unlabeled data classification. *Optimization Methods and Software*, 15 :29–44, 2001.

[36] Carl Gold and Peter Sollich. Fast bayesian support vector machine parameter tuning with the nystrom method. In *IJNN'05*, pages 2820 – 2825, 2005.

[37] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 529–536. MIT Press, Cambridge, MA, 2004.

[38] Yann Guermeur, André Elisseeff, and Hélène Paugam-Moisy. A new multi-class SVM based on a uniform convergence result. volume 4, page 4183, 2000.

[39] Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. In *NIPS*, 1997.

[40] J. H. Holland. *Adaptation in Natural and Artificial Systems : An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* University of Michigan Press, 1975.

[41] J. H. Holland. *Adaptation in Natural and Artificial Systems : An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, 2nd ed.* Cambridge, MA : MIT Press, 1992.

[42] C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, 2001.

[43] T.S. Jaakkola and D. Haussler. Probabilistic kernel regression models. *Workshop in Conference on Artificial Intelligence and Statistics*, 1999.

[44] Licheng Jiao, Liefeng Bo, and Ling Wang. Fast sparse approximation for least square support vector machine. *IEEE Transactions on Neural Networks*, 18(3) :685–697, 2007.

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

[46] T. Joachims. Estimating the generalization performance of a SVM efficiently. *International Conference on Machine Learning*, pages 431–438, 2000.

[47] T. Joachims. Transductive learning via spectral graph partitioning. In *Proceeding of The Twentieth International Conference on Machine Learning (ICML-2003)*., 2003.

[48] Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

[49] S. Thrun K. Nigam, A. McCallum and T. Mitchell. Text classification from labeled and unlabeled documents using em. *machine Learning*, 39(2/3) :103–134, 2000.

[50] S.S. Keerthi and S.K. Shevade. Smo algorithm for least-squares SVM formulations. *Neural Computation*, 15 :487–507, 2003.

[51] 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*, 13 :637–649, 2001.

[52] Griffiths T. L. Stromsten S. Kemp, C. and J. B. Tenenbaum. Semi-supervised learning with trees. In *NIPS*, 2003.

[53] Julia A. Lasserre, Christopher M. Bishop, and Thomas P. Minka. Principled hybrids of generative and discriminative models. In *CVPR '06 : Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 87–94, Washington, DC, USA, 2006. IEEE Computer Society.

[54] J.I Marden. Hypothesis testing : from p values to bayes factors. *Journal of the American Statistical Association*, 95 :1316 – 1319, 2000.

[55] Patrick Pakyan Choi John Lafferty Brian Pantano Mugizi R. Rwebangira Xiaojin Zhu Maria-Florina Balcan, Avrim Blum. Person identification in webcam images : An application of semi-supervised learning. 2005.

[56] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, A209 :415–446, 1909.

[57] T. Mitchell. The role of unlabeled data in supervised learning. In *Proceedings of the Sixth International Colloquium on Cognitive Science, San Sebastian, Spain.*, 1999.

[58] M. Moreira and E. Mayoraz. Improved pairwise coupling classification with correcting classifiers. In *ECML*, pages 160–171, 1998.

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

[60] Manfred Opper and Ole Winther. Mean field methods for classification with gaussian processes. In *the 1998 conference on Advances in neural information processing systems II*, pages 309–315. MIT Press, 1999.

[61] E. Osuna, R. Freund, and F. Girosi. Improved training algorithm for support vector machines. In *NNSP'97*, 1997.

[62] J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A.J. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, editors, *Advances in Large Margin Classiers*, pages 61–74. 2000.

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

[64] Joel Ratsaby and Santosh S. Venkatesh. Learning from a mixture of labeled and unlabeled examples with parametric side information. In *Computational Learing Theory*,

pages 412–417, 1995.

[65] Wiebe J. Riloff, E. and T. Wilson. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the Seventh Conference on Natural Language Learning*, 2003.

[66] Schneiderman H. Rosenberg C., Hebert M. Semi-supervised self-training of object detection models. In *WACV 2005.*, 2005.

[67] B. Scholkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[68] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[69] Chapelle O. Sindhwani V., Keerthi S.S. Deterministic annealing for semi-supervised kernel machines. In *International Conference on Machine Learning (ICML)*, 2006.

[70] Peter Sollich. Probabilistic methods for support vector machines. In *Conference on Advances in neural information processing systems 12*, pages 349–355. MIT Press, 2000.

[71] J. A. K. Suykens, L. Lukas, and L. Vandewalle. Sparse least squares support vector machine classifiers. In *European Symposium of Artificial Neural Networks*, 2000.

[72] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

[73] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3) :293–300, 1999.

[74] Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[75] M. Meila T. Jaakkola and T. Jebara. Maximum entropy discrimination. In *In Advances in Neural Information Processing Systems 12. MIT Press*, 1999.

[76] I. Ulusoy and C. M. Bishop. Generative versus discriminative models for object recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition*, San Diego, 2005.

[77] T. Van Gestel, J. A. K. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and J. Vandewalle. Benchmarking least squares support vector machine classifiers. *Machine Learning*, 54(1) :5–32, 2004.

[78] Lanckriet G. Lambrechts A. De Moor B. Vandewalle J. Van Gestel T., Suykens J. Bayesian framework for least squares support vector machine classifiers, gaussian processes and kernel fisher discriminant analysis. *Neural Computation*, 15(5) :1115–

1148, 2002.

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

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

[81] V. N. Vapnik. Principles of risk minimization for learning theory. *Adavances in Neural Information Processing Systems 4, Morgan Kaufman, San Mateo, CA*, pages 831–838, 1992.

[82] V. N. Vapnik, O. Chapelle, and J. Weston. Transductive inference for estimating values of functions. *Adavances in Neural Information Processing*, 1999.

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

[84] G. Wahba, Y. Lin, and H. Zhang. Generalized approximate cross validation for support vector machines, or, another way to look at margin-like quantities. Technical report, Departement of Statistics,University of Wisconsin, February 25 1999.

[85] Fei Wang and Changshui Zhang. Label propagation through linear neighborhoods. *Knowledge and Data Engineering, IEEE Transactions on*, 20(1) :55–67, Jan. 2008.

[86] Ye Wang and Shang-Teng Huang. Training TSVM with the proper number of positive samples. *Pattern Recognition Letters*, 26(14) :2187–2194, 2005.

[87] Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5 :975–1005, 2004.

[88] Guoping Wang Yisong Chen and Shihai Dong. Learning with progressive transductive support vector machine. *Pattern Recognition Letters*, 24(12) :1845–1855, 2003.

[89] Chen lin Yuangui Li and Weidong Zhang. Improved sparse least-squares support vector machine classifiers. *Neurocomputing*, 69 :1655–1658, 2006.

[90] Song-Feng Zheng. Least square support vector machine and its bayesian interpretation. Technical report, Technical Report, June 2004.

[91] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. 2003. In 18th Annual Conf. on Neural Information Processing Systems.

[92] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2008.

[93] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *In ICML*, pages 912–919, 2003.

[94] G. Zoutendjik. Methods of feasible directions : astudy in linear and non-linear programming. 1970.

ANNEXES

Dans cette partie, nous avons annexé les articles suivants avec la mise en page du manuscrit soumis aux journaux.

[1] Adankon, M.M., and Cheriet, M. (2009). "Genetic Algorithm based training for Semi-supervised SVM". Soumis à Neural Computing and Applications.

[2] Adankon, M.M., Cheriet, M, and Biem, A. (2009), "Semi-Supervised Least Squares Support Vector Machine". Sous presse à IEEE Transactions on Neural Networks.

[3] Adankon, M.M., Cheriet, M, and Biem, A. (2009), "Semi-Supervised Learning using Bayesian inference". Soumis à IEEE Transactions on Pattern Analysis and Machine Intelligence.

[4] Adankon, M.M., and Cheriet, M. (2009)."Help-Training for Semi-supervised Learning". Soumis à Pattern Recognition Journal.

[5] Adankon, M.M., and Cheriet, M. (2009). "Model Selection for LS-SVM : Application to Handwriting Recognition". Pattern Recognition 42(12) : 3264-3270.

ARTICLE 1

GENETIC ALGORITHM BASED TRAINING FOR SEMI-SUPERVISED SVM

M. M. Adankon[1], and M. Cheriet[1],
[1]École de Technologie Supérieure, 1100 Notre-Dame Ouest,
Montréal, Québec, Canada H3C 1K3

Cet article est soumis à Neural Computing and Applications.

Editorial Manager(tm) for Neural Computing and Applications
Manuscript Draft

Manuscript Number:

Title: Genetic Algorithm based training for Semi-supervised SVM

Article Type: Original Article

Keywords: semi-supervised learning, svm, support vector machine, genetic algorithm

Corresponding Author: Mathias Adankon,

Corresponding Author's Institution: University of Quebec /ETS

First Author: Mathias Adankon

Order of Authors: Mathias Adankon; Mohamed Cheriet

Neural Computing and Applications manuscript No.
(will be inserted by the editor)

Genetic Algorithm based training for Semi-supervised SVM

Mathias M. Adankon · Mohamed Cheriet

Received: date / Accepted: date

Abstract The Support Vector Machine (SVM) is an interesting classifier with excellent power of generalization. In this paper, we consider applying the SVM to semi-supervised learning. We propose using an additional criterion with the standard formulation of the semi-supervised SVM (S^3VM) to reinforce classifier regularization. Since, we deal with non-convex and combinatorial problem, we use a genetic algorithm to optimize the objective function. Furthermore, we design the specific genetic operators and certain heuristics in order to improve the optimization task. We tested our algorithm on both artificial and real data, and found that it gives promising results in comparison with classical optimization techniques proposed in literature.

Keywords semi-supervised learning · genetic algorithm · support vector machine · SVM.

1 Introduction

Pattern recognition problems are solved using classifiers which are machines built by using prototypes of the data to be recognized. Traditionally, machine learning needs labeled data, as does the training procedure called supervised learning. Now, labeling, as well as the collection of the data itself, is a process which requires considerable human effort and is therefore very expensive. The labeling of handwritten documents, images or web pages, for example, requires both human expertise and insight while in the field of medicine or biology, testing and very complex experiments are sometimes needed for this process and in certain cases, it may be very difficult or even impossible to label all the available data.

This work is extended version of the IJCNN2007 paper, where we include the following important new information: a thorough explanation and analysis of the objective function, use of the connectivity concept for selecting the unlabeled subset during the optimization process and improved experimental results.

Mathias M. Adankon
Synchromedia Laboratory, École de Technologie Supérieure, University of Quebec,
1100 Notre-Dame West, Montreal, Canada, H3C 1K3.
E-mail: mathias.adankon@synchromedia.ca

Mohamed Cheriet
E-mail: mohamed.cheriet@etsmtl.ca

Since we know that the larger the number of training samples, the better the performance of the classifier -making assumption that we have a correct classifier model-, the issue becomes finding a way to improve supervised learning by adding unlabeled data, or determining what information can be obtained from unlabeled data to improve supervised learning [24]. Training using both labeled and unlabeled data is called semi-supervised learning [33, 27]. In this type of training, the classifier is built by learning from both labeled and unlabeled data. Hence, it is not necessary to label all the data collected in order to train the classifier.

Support vector machines (SVMs) are particular linear classifiers which are based on the margin-maximization principle. They use the kernel trick, a technique recently introduced in the machine learning community, to produce nonlinear boundaries. The idea behind kernels is to map training data nonlinearly into a higher-dimensional feature space via a mapping function Φ and to construct a separating hyperplane that maximizes the margin. The construction of the linear decision surface in this feature space only requires the evaluation of the dot product $\Phi(x).\Phi(y) = k(x, y)$, where $k()$ is called the kernel function [26].

SVMs perform structural risk minimization, which was introduced to machine learning by Vapnik [31], and they have yielded excellent generalization performance. Many applications have been developed successfully with SVMs. The idea of using this classifier in a semi-supervised context is therefore an appealing one, and as a result, we can design a classifier for applications where a large amount of data is available without labeling all that data.

Semi-supervised SVM (S^3VM) is based on the following idea: margin-maximization on both the labeled and unlabeled data. This approach is introduced first by Vapnik [31] for formulating Transductive SVM (TSVM). It implements the Cluster assumption for semi-supervised learning: if points are in the same cluster, they are likely to be of the same class; in other words, objects of two distinct classes can not be found in the same cluster [8]. However, there are two problems associated with the S^3VM standard formulation:
- Margin-maximization on unlabeled data does not directly reinforce the regularization of the classifier, but rather implements the cluster assumption [12].
- Unlike the SVM, the optimization problem resulting from the S^3VM is not convex. This makes it difficult to find the optimal solution, whereas an optimization algorithm like SVM-light will sometimes converges to a local minimum [22].

In order to overcome these problems, we propose in this paper to :
(i) reinforce the regularization of the classifier by penalizing the complexity of the machine and,
(ii) resolve the optimization problem using a genetic algorithm (GA) [14] based on the combinatorial view of the problem.

Evolutionary algorithms have been applied successfully with SVM classifier for many tasks such as feature selection, model selection(turning hyperparameters), multiclass problem, transductive learning, etc. [15, 16, 28, 1, 23]. In this paper, we focus on semi-supervised training of SVM where we reformulate a new objective function and use GA for performing the optimization task with certain heuristics in order to avoid trapping into local optimum. Then, the resulting classifier is improved.

This paper is structured as follows. In section 2, we conduct a review of the S^3VM. In section 3, we discuss the regularization reinforcement of the S^3VM. In section 4, we describe a GA for training a semi-supervised SVM. In section 5, we describe the implementation strategy. In section 6, we present the experimental results on both artificial and real data. We conclude the paper in section 7.

2 Semi-Supervised Support Vector Machine

We first consider a binary classification problem. Let us begin with labeled data $\{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}$ and unlabeled data $\{x_1^*, \ldots, x_n^*\}$ with $x_i \in \mathbb{R}^d$, $x_i^* \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$.

In supervised learning, the training algorithm of the SVM resolves the following optimization problem which expresses the maximization of the margin $2/\|w\|$ and the minimization of the training error.

$$\min_{w,b,\xi} \frac{1}{2} w'w + C \sum_{i=1}^{\ell} \xi_i \tag{1}$$

$$\text{s.t. } : y_i[w'\phi(x) + b] \geq 1 - \xi_i \quad \forall i = 1, ..., \ell \tag{2}$$

$$\xi_i \geq 0 \quad \forall i = 1, ..., \ell \tag{3}$$

In the primal form, the Lagrangian of this problem is as follows:

$$L = \frac{1}{2} w'w + C \sum_{i=1}^{\ell} \xi_i - \sum_{i=1}^{\ell} \alpha_i [y_i (w'\phi(x) + b) - 1 + \xi_i] - \sum_{i=1}^{\ell} \lambda_i \xi_i \tag{4}$$

with the Lagrange multipliers $\alpha_i \geq 0$ and $\lambda_i \geq 0$ for all $i = 1, ..., \ell$.

When, we apply the differentiation theorem w.r.t. Lagrange, we obtain :

$$f(x) = w'\phi(x) + b = \sum_{i=1}^{\ell} \alpha_i y_i k(x_i, x) + b \tag{5}$$

with α solution of :

$$maximize : W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{6}$$

$$s.t. : \sum_{i=1}^{\ell} \alpha_i y_i = 0 \quad and \quad 0 \leq \alpha_i \leq C, i = 1, ..., \ell$$

In the case of semi-supervised learning, S^3VM tries to find the hyperplane $< w, b >$ and the labels $y_1^*, \ldots, y_n^*$ of the unlabeled data that maximize the margin with minimum error.

$$\min_{w,b,\xi,\xi^*,y_1^*,\ldots,y_n^*} \mathcal{L} = \frac{1}{2}w'w + C\sum_{i=1}^{\ell}\xi_i + C^*\sum_{j=1}^{n}\xi_j^* \tag{7}$$

$$\text{s.t.} : y_i[w'\phi(x_i) + b] \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i = 1, ..., \ell \tag{8}$$

$$y_j^*[w'\phi(x_j^*) + b] \geq 1 - \xi_j^*, \quad \xi_j^* \geq 0 \quad \forall j = 1, ..., n \tag{9}$$

Vapnik [31] proposed this formulation to reinforce the classifier in the test by adding the minimization of the error on the test set $\{x_1^*, \ldots, x_n^*\}$. We note that equation (7) minimizes both the training error and the test error. Solving the problem expressed by equation (7) is then equivalent to finding the hyperplane that separates both the training and test data with the minimum cost error.

Using the combinatorial optimization approach, the goal is to find first the set of binary variables. And for the fixed $y_1^*, \ldots, y_n^*$ we obtain the following functional, which must be maximized :

$$\begin{aligned} W_{y_1^*,\ldots,y_n^*}(\alpha, \alpha^*) = &\sum_{i=1}^{\ell}\alpha_i - \frac{1}{2}\sum_{i,r=1}^{\ell}\alpha_i\alpha_r y_i y_r k(x_i, x_r) \\ &+ \sum_{j=1}^{n}\alpha_j^* - \frac{1}{2}\sum_{j,r=1}^{n}\alpha_j^*\alpha_r^* y_j^* y_r^* k(x_j^*, x_r^*) \\ &- \sum_{i=1}^{\ell}\sum_{j=1}^{n}\alpha_i\alpha_j^* y_i y_j^* k(x_i, x_j^*) \end{aligned}$$

$$\begin{aligned} subject \quad to: \\ &0 \leq \alpha_i \leq C, i = 1, ..., \ell \\ &0 \leq \alpha_j^* \leq C^*, j = 1, ..., n \\ &\sum_{i=1}^{\ell}\alpha_i y_i + \sum_{i=j}^{n}\alpha_j^* y_j^* = 0. \end{aligned} \tag{10}$$

Optimizing the problem of equation (10) is very difficult. Finding the exact solution requires searching over all possible 2^n labels of the unlabeled data. We can do this when we have a small number of unlabeled data but, for a large number of them, it is just not feasible.

In 1998, Bennett and Demiriz [6] were the first to implement the S^3VM by using an integer programming method. However, their method is intractable for large problems. One year later, Joachims [22] proposed a combinatorial approach to solving the S^3VM problem with certain heuristics, the implementation of which solves large problems however, convergence is still not guaranteed. Recently, Chapelle et al.[9] proposed to apply branch and bound techniques for obtaining globally optimal solution. While their method gives a good solution, its implementation does not scale to a large dataset. In the same year Collobert et al.[13] proposed $S^3VM - CCCP$ (concave-convex procedure), which decomposes a non-convex loss function into a convex part and a concave part, and where this last part is iteratively approximated by a Taylor expansion. Other techniques used to optimize the original S^3VM problem eq. (7)-(9) can be found in [10].

3 Regularization reinforcement of the semi-supervised SVM

In machine learning, regularization is very important in improving generalization of the machine. The margin-maximization principle provides a good power of generalization to the supervised SVM. However, using this principle with unlabeled data is more a matter of implementing the cluster assumption than it is of regularization. As a result, regularization of the S^3VM becomes weaker than the supervised SVM and needs to be reinforced.

We consider here an example which shows the weakness of the expression of the original S^3VM in terms of generalization; specifically a two-class problem, where each class is generated with equal probability from a Gaussian distribution in 2D[1]. Figure 1 illustrates the problem where we have labeled and unlabeled samples. Figures 2-a and 2-b show two solutions for the semi-supervised problem using a SVM classifier. The solution represented in Figure 2-a is optimal when we consider the real distribution of the data, with $\mathcal{L} = 84.33$ and the number of support vectors equal to 13, while the objective function of the solution in Figure 2-b is $\mathcal{L} = 79.43$ with 16 support vectors. We note that the previous value of the S^3VM objective function is better than the value of the optimal solution. Hence, when we minimize only the expression $\mathcal{L}$ in (7), we do not necessarily find the optimal solution, but in this case we obtain a solution with a weak regularization.

The previous example shows what may happen when we use only the original S^3VM formulation to perform the semi-supervised SVM. Clearly, it is necessary to reinforce the regularization of the machine. In his book [31], Vapnik mentioned that *"the generalization ability of the constructed hyperplane depends on the number of support vectors"*. So, we propose to use that number to reinforce regularization. This criterion not only penalizes the complexity of the machine but it is also simple and does not need much computational time.

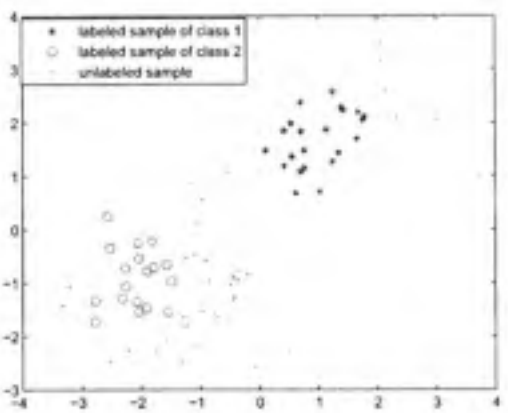
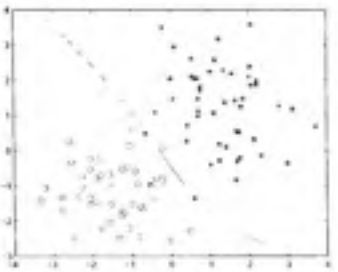
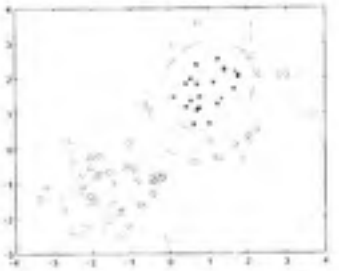


Fig. 1 Labeled and unlabeled samples

[1] The problem described in Figure 2 is just a visual example of the introduction of the regularization idea. In addition, it is important to state that we use the same kernel in Figures 2-a and 2-b. Also, we have repeated the example with other kernels with the same result.



(a) The optimal solution for the semi-supervised problem presented in Figure 1, according to the distribution of the data.



(b) Solution for the semi-supervised problem presented in figure 1 where the S^3VM objective function value is better than the optimal solution found in (a).

Fig. 2 The figures show two solutions for the problem presented in Figure 1 and the bound decision in each case.

3.1 Our objective function

The objective function of the original S^3VM is given by :

$$\mathcal{L}(w, b, y_1^*, ..., y_n^*) = \frac{1}{2}\|w\|^2 + C\sum H(y_i[w'\phi(x_i)+b]) + C^*\sum H(y_j^*[w'\phi(x_j^*)+b]) \quad (11)$$

where function $H(t) = max(0, 1-t)$ is the Hinge Loss.

Since our aim is to reinforce regularization, we add the criterion that penalizes the complexity of the classifier which consists in reducing the number of support vectors [31, 11]. Hence, we propose the following objective function :

$$\mathcal{F} = \beta\mathcal{L}(w, b, y_1^*, ..., y_n^*) + (1-\beta)N_{VS} \quad (12)$$

where N_{VS} represents the number of support vectors and β is a real number with $0 \leq \beta \leq 1$.

This objective function combines the original S^3VM formulation with an extra criterion, the task of which is to regularize the machine. Setting β equal to 1 eliminates this regularization term in (12), while setting it equal to 0 eliminates the term of the standard S^3VM. Considering that the first term of (12) is the main term, we must not use a β value of less than 0.5. However, using a β value near 1 is not good, because in doing so we reduce the contribution of the second term. Based on empirical results, we suggest setting the β value between 0.6 and 0.7. We in fact used a β value equal to 0.65 in this paper.

Sometimes, when the labeled set is too small, we need to use the balancing constraint in order to avoid unbalanced solutions. Without this constraint, the algorithm classifies all the unlabeled samples in the same class when $n \ll \ell$. In this case, we use the following constraint which represents the ratio of the positive samples in the unlabeled set :

$$R = \frac{1}{2}\left[1 + \frac{1}{n}\sum_{j=1}^{n} y_j^*\right] \tag{13}$$

This constraint was first used by Joachims [22]. Since the true value of R is unknown, it is estimated from the labeled set or from prior knowledge about the problem.

4 Genetic Algorithm for training the semi-supervised SVM

S^3VM formulation produces a non-convex optimization problem. Considering the combinatorial approach for this problem, we propose to use a GA [20, 21] which works very well on combinatorial problem.

The GA is a particular case of Evolutionary Algorithms which are based on biological phenomena [4]. Specifically, the GA imitates the operations produced during the reproduction of the species. The GA, unlike the classical optimization methods, does a parallel search, and it has the advantage of adapting to complex problems where derivability or convexity is not required. Also, the continuity of the search space is not a constraint for the GA, as in our case, the labels of the unlabeled data have a discrete value. Furthermore, the GA is of interest for solving the optimization problem where the search space is very large.

The strategy used to learn the semi-supervised SVM is represented in Figure 3. We use a standard algorithm to train the SVM and a GA to search for the best labels.

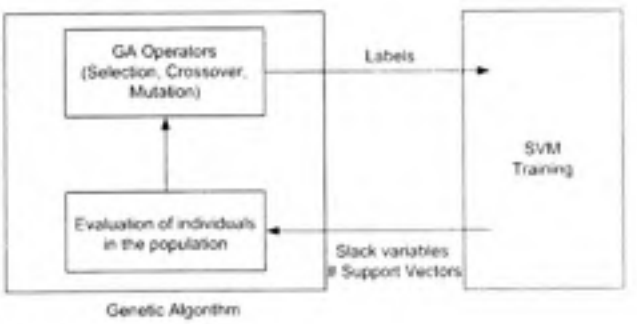


Fig. 3 Learning semi-supervised SVM with GA.

4.1 Representation

In order to use the GA to find the optimal labels $y_1^*, ..., y_n^*$, that contribute both to maximizing the margin and minimizing the error, we first need to represent them in a binary

string. In classification with SVM, the labels belong to $\{1, -1\}$, and the binary representation is sufficient. Classically, the binary representation of the variables uses the bits 0 and 1; but in our case, we use the values -1 and 1 directly. For example, if we have a problem with 7 unlabeled samples, an individual solution of the population can be represented by $(1; -1; -1; 1; 1; 1; -1)$ or $(-1; 1; -1; 1; 1; -1; 1)$.

4.2 Evaluation

This step of the GA is necessary in order to determine the quality of each candidate solution in the population. In this case, we compute the value of the objective function defined in equation (12).

4.3 Selection

The selection operator of the GA performs the following tasks: identify good solutions in a population, make multiple copies of the good solutions and eliminate bad solutions from the population. There are several techniques for carrying out the selection, and research has been conducted to evaluate the performance of these techniques [17, 29]. They all have some advantages and disadvantages. For our problem, we have chosen the selection method called the *Tournament selection* [3, 17], and we use the heuristic that enables us to select the best solution at each iteration (generation). This is because, when we use the Tournament selection technique, we do not have any guarantee that the best solution has been selected at that moment. Hence, we first select the best solution in the current population at each moment and, second, we complete the selection using the Tournament technique.

4.4 Crossover

The crossover operator is one of the two GA operators that create the new solutions in the population. Crossover is the process of combining the genes of one individual with those of another to create the new individuals that inherit characteristics of both parents. Like the selection operator, there are a number of crossover operators in the GA literature [29]. We use the two- and three-points crossover techniques according to the chromosome length, which is dependent on the size of the unlabeled set (Figure 4).

4.5 Mutation

In general, when this operator is applied on a chromosome (individual), a randomly selected gene is changed into its opposite. Then, a new individual is created in the population. For our problem, we use the criterion that makes it possible to mutate only the misclassified samples according to the margin. This is why the probability of selecting a gene is set proportional to the corresponding slack variable. Hence, the mutation probability of the genes corresponding to the non support vector is zero, since ξ_j is equal to zero for non support

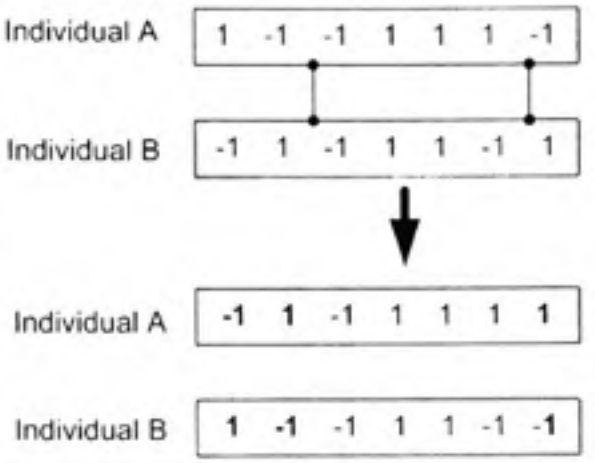


Fig. 4 Example of the crossover operator. In this figure, the labels of the unlabeled samples are represented by two individuals (solutions) before and after the two-points crossover technique has been applied.

vectors. Thus, we propose to express the mutation probability on a gene as follows:

$$P_m = P_0 * \frac{\xi_j}{\sum \xi_i} \tag{14}$$

where P_0 is the probability of choosing the individual that possesses the gene.

We note that the mutation does not occur when the corresponding sample is well classified and located away from the margin ($\xi_j = 0$). With this heuristic, we are prevented from changing the labels of samples that are well classified with a high degree of confidence (Figure 5).

5 Implementation Strategy

In order to improve convergence time, we begin the optimization process with a subset of the unlabeled data, and during the procedure we add a part of the remaining samples.

First, we train the classifier only with the labeled data, and with this preliminary machine, we label the unlabeled data. When the output of the SVM is large, $abs[f(x_i)] > 1$, the label is kept, otherwise the label is chosen randomly. Hence, we initialize the population close to the solution. This heuristic enables us to guide the initialization of the population.

Second, we use a subset of the unlabeled samples to start the optimization process. Then, the search space is reduced and the convergence is accelerated. The idea behind this technique is to start the optimization in a subspace, and to enlarge that space at each step. The details are shown in Figure 6, and it is inspired by the strategy we developed in [2]. In section 5.1, we describe the way to select the unlabeled subset in order to accelerate the

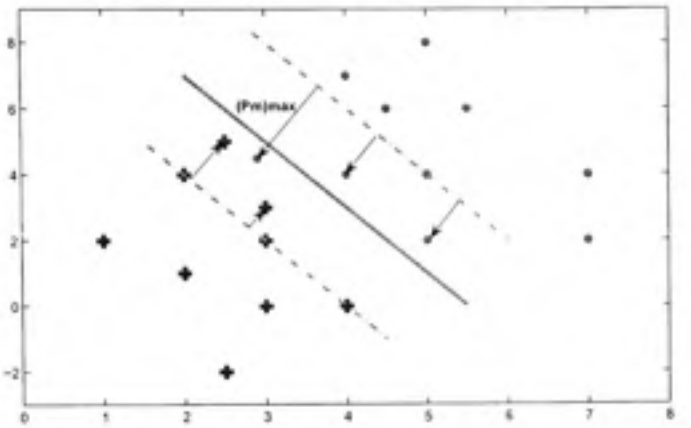


Fig. 5 Illustration of the mutation probability behavior with respect to the corresponding slack variable. In the figure, we indicate the sample with the high probability P_m of being mutated into its opposite.

convergence of the algorithm.

The stopping criterion that we use is the number of iterations. We may also check the fitness value and stop the algorithm when the best fitness value has not been improved after a specified number of iterations.

The various heuristics used for initializing the population, designing the genetic operators and controlling the GA parameters are based on a structured approach to representing domain knowledge, as described in [25]. This approach allows the use, both implicitly and explicitly, of the knowledge we have on the optimization problem.

5.1 How to select the unlabeled subset?

We can randomly choose the samples forming the current unlabeled subset, but, in our case, we use a heuristic which enables us to keep the continuity and the compactness of the current working set. We use the distance criterion to define the proximity of the samples. Then, at each iteration, the unlabeled samples nearest to the previous working set are selected and added to the current unlabeled subset. With this technique, we permit the connectivity of the working set samples and avoid the creation of low densities where none existed before. This is because semi-supervised learning is based on the cluster assumption, and the creation of the low densities in the structure of the data has a negative impact on the classification decision boundary. The different heuristics described here empowered our algorithm and help it to avoid falling into local optimum.

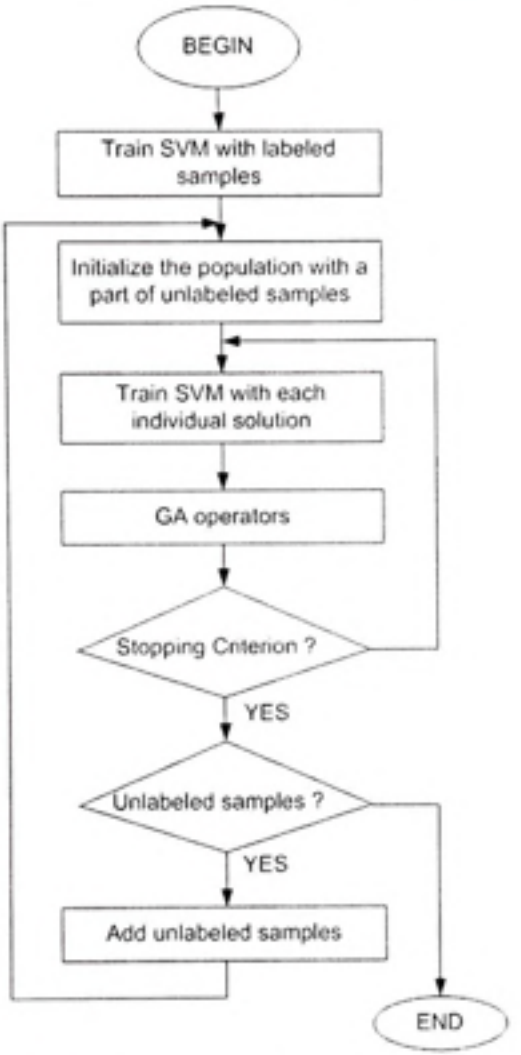


Fig. 6 Genetic algorithm based training for Semi-supervised SVM.

6 Experiments

First, we consider the SVM in supervised learning. The hyperparameters are selected by cross-validation, and then we use these values in semi-supervised learning as optimal values of the Gaussian kernel width and hyperparameters C. Also, we make the assumption that, during the optimization process, the labels found for the unlabeled samples are right. So, we set $C^* = C$.

6.1 Datasets

6.1.1 Gaus50

We consider a two-class problem in a 50-dimensional input space like the artificial datasets used in [18]. The data of each class are generated with equal probability from a Gaussian distribution with a unit covariance matrix. The mean of the Gaussian is $(a, a, ..., a)$ for class 1 and $-(a, a, ..., a)$ for the class 2. In order to obtain a Bayes error equal to 5%, the parameter a is fixed to 0.23. We generated 550 samples for training and 1000 samples for testing.

6.1.2 Gaus50x

We have here another artificial two-class problem in a 50-D space but with multimodal distribution. The data of each class are generated with equal probability from a Gaussian mixture distribution with significant overlap. The two classes have equal prior and their class-conditional distributions are respectively $p(x|y = 1) = 0.49\mathcal{N}(\mu_1, I) + 0.51\mathcal{N}(\mu_2, I)$ and $p(x|y = -1) = 0.49\mathcal{N}(-\mu_1, I) + 0.51\mathcal{N}(-\mu_2, I)$ where $\mu_1 = (0.25, 0.25, ..., 0.25, 0.25)$, $\mu_2 = (0.25, 0.25, ..., -0.25, -0.25)$, $N(\mu, I)$ is a Gaussian density function with its mean vector μ, and I is the identity matrix for its covariance matrix.

6.1.3 Text

Text document classification is a problem involving high dimensionality. We used the *mac* and *windows* classes taken from the 20 news group dataset, with 7511 dimensions, as pre-processed in [30]. We divided the dataset in two subsets : 1446 samples for training and 500 for testing.

6.1.4 USPS

It is the well known US Postal Service handwritten digits recognition corpus. The digits are represented by normalized grey scale images of size 16×16. The learning dataset contains 7291 samples for training while the testing dataset consists of 2007 other samples. We have multi-class problem with 10 classes. We then trained 10 machines using the *one-against-all* strategy.

6.2 Accuracy of our algorithm

In this section, we tested our method and compared it with $S^3VM - Light$ which is the first public implementation of S^3VM, under the name TSVM in popular *SVMlight* software. We perform this set of experiments on Gaus50 dataset in order to illustrate how our algorithm based on GA and certain heuristics can avoid trapping into local optimum.

We randomly selected 50 samples from the training set to form the labeled data and the remaining data (500 samples) consisting of unlabeled samples. We repeated this operation 10 times and at each time, we tested *SVMlight* and our semi-supervised algorithm. We ran

our algorithm 30 times for each problem because a genetic algorithm is a stochastic process. The results are shown in Table 1 where we report the mean test error rate and the standard deviation obtained by our algorithm for each selection. We used the Gaussian kernel with the following hyperparameters for SVM/S^3VM learning $\gamma = 1/\sigma^2 = 0.02$ and $C = C^* = 1$.

The results presented in Table 1 show that *SVMlight* sometimes converges to a local minimum, in which case, the algorithm gives bad results. For example, in the case of selection #9, *SVMlight* gives a critical error rate of 32.4%. At the same time, we can note that our algorithm gives promising results for all 10 selections. In general, it performs better than *SVMlight* and it has capacity to escape some local minima, e.g. selection #9. Moreover, the accuracy of the classifier designed with semi-supervised learning is improved. Since, when we knew the real labels of the unlabeled samples and we trained supervised SVM, we obtained 5.9% error rate that is approximatively found by our semi-supervised algorithm.

Table 1 Test error rate obtained with our algorithm and SVMlight; when we trained the supervised SVM with all the available data 550 samples as labeled, we obtained 5.9% error rate on test set.

Selection	Our algorithm		SVMlight
	Error rate Mean	Standard deviation	
#1	5.92%	0.29%	6.8%
#2	7.17%	0.56%	13.6%
#3	6.38%	0.56%	8.8%
#4	6.14%	0.10%	6.3%
#5	7.08%	0.25%	8.8%
#6	6.26%	0.66%	6.4%
#7	7.08%	0.60%	7.6%
#8	7.52%	0.86%	8.8%
#9	6.61%	0.43%	32.4%
#10	5.85%	0.41%	5.8%

6.3 Impact of the regularization term

Adding a regularization term in the form of a number of support vectors reinforces the generalization performance of the classifier. In this section, we perform experiments on data used in the previous section with the same setup, in order to show the contribution of this term. We train the standard S^3VM with GA by setting β to 1 without the constraint (13) and repeat the experiment with the constraint (13). The results are plotted in Figure 7.

It is remarkable that:
(1) the additional term designed to reinforce the regularization improves the performance of the classifier; and
(2) the contribution of the constraint (13) which is similar to those used in other semi-supervised methods [22, 9] is not clearly apparent in this case.

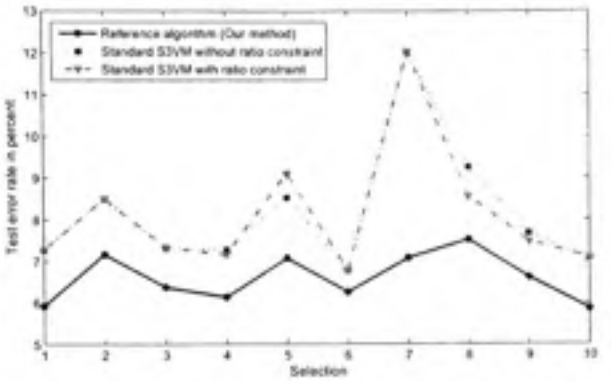


Fig. 7 Impact of the regularization term on the performance of the classifier: the reference algorithm represents our method with this regularization, using $\beta = 0.65$.

6.4 Comparison with other semi-supervised SVM algorithms

It would be interesting to compare our GA learning algorithm with other S^3VM methods from the literature as well, such as ∇S^3VM [12], $CCCP$ [13], and *Branch and Bound* [9]. The first method, ∇S^3VM, was proposed to optimize the S^3VM objective function (which places the decision boundary in low density regions) by using gradient descent process. The second, $CCCP$, applies the concave-convex procedure to the S^3VM optimization problem where non-convex function is decomposed into a convex component and a concave component and at each iteration, the concave part is replaced by a approximation term. The third, *Branch and Bound*, is designed to find exact and globally optimal solutions with high computational cost.

First, we compared these various algorithms on the **Two Moons** dataset, which is a standard benchmark for the semi-supervised learning algorithms used in the literature [10, 19, 7, 5, 9, 32]. We used the same setup and the same hyperparameters($\gamma = 2$ and $C = C^* = 10$) as in [9] and the results are shown in table 2. The reported test errors obtained by ∇S^3VM, $S^3VM - CCCP$, $S^3VM - Light$ and *Branch and Bound* are taken from [10], where its authors claim that their technique, *"Branch and Bound"*, provides a globally optimal solution. However, their method does not scale to large datasets. Our algorithm achieves the same error rate as *"Branch and Bound"*, but does perform on large datasets.

Table 2 Test error rate on the Two Moons dataset

	Test error rate
∇S^3VM	61%
$S^3VM - CCCP$	63%
$S^3VM - Light$	68%
Branch and Bound	0%
Our method	0%

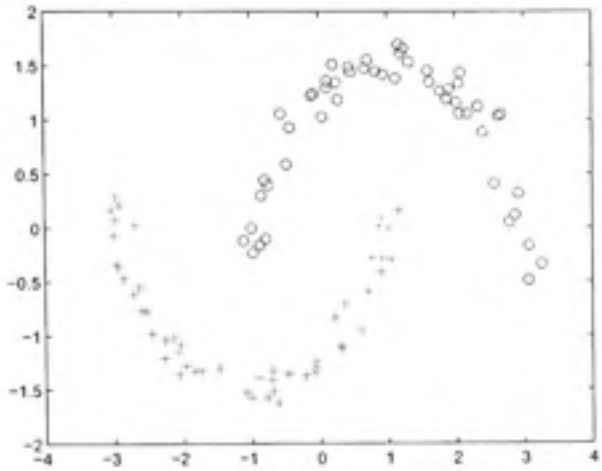


Fig. 8 Illustration of Two Moons problem

As mentioned by the authors, Branch and Bound implementation does not scale to large datasets. Also, ∇S^3VM is implemented to return the error on an unlabeled dataset but not on a test set. So, only $S^3VM - CCCP$ and $S^3VM - Light$ algorithms, which codes are available on website, is tested on the datasets described in section 6.1. For each dataset, we labeled 10% of the training samples and kept the rest as unlabeled data. The error rate is computed on the test set different from unlabeled training set. Table 3 provides the values of hyperparameters with respect to each dataset. The results obtained after running the algorithm are summarized in Table 4. We note that our algorithm performs better than other S^3VM algorithms for two-class problems, although $S^3VM - CCCP$ algorithm and ours achieved approximatively the same result on a multi-class problem. And we think that the result obtained with multi-class problem may be due to the multi-class coupling strategy we used to combine the SVM outputs, which is different from the optimization task. Figures (9) and (10) show the variation of the test error when we increased the number of labeled samples. We see that our method outperforms other S^3VM algorithm when we have few labeled samples.

Table 3 Hyperparameter values used for our S^3VM algorithm.

	Kernel parameter $\gamma = 1/\sigma^2$	Hyperparameter $C = C*$
Gaus50	0.02	1
Gaus50x	0.01	1
Text	0.02	10
USPS	0.0078	10

Finally, the various experiments illustrate the performance of our algorithm. This performance improvement was mainly due to the following:

Table 4 Comparison between our semi-supervised SVM algorithm ($S^3VM - GA$), $S^3VM - light$ [22] and $S^3VM - CCCP$ [13]. In the table we report error rate on test set.

	$S^3VM - light$	$S^3VM - CCCP$	$S^3VM - GA$ (Our method)
Gaus50	8.80%	7.70%	**6.30%**
Gaus50x	16.30%	15.50%	**14.70%**
Text	5.40%	5.45%	**4.53%**
USPS	14.70%	**7.32%**	7.62%

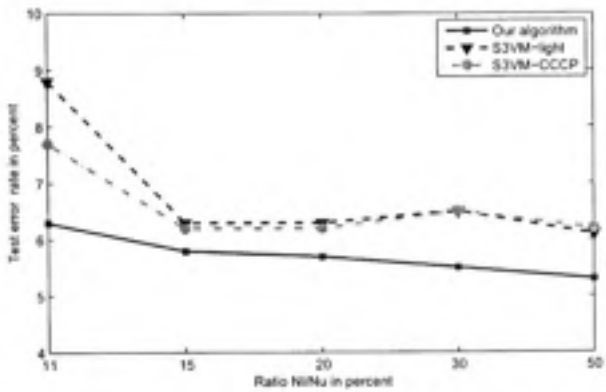


Fig. 9 Behavior of our algorithm with respect to the ratio N_l/N_u on Gaus50 dataset, where N_l represents the number of labeled samples and N_u the number of unlabeled samples.

- We added a penalty term to the S^3VM objective function, which controls the complexity of the model. In fact, the generalization performance of the classifier designed is reinforced.
- We used a genetic algorithm which performs better than classical optimization algorithms for the combinatorial problems and some heuristics to avoid falling into local optimum.

These two important properties of our algorithm provide an advantage over classical S^3VM algorithms. We should mention, however, that ours carries a heavy computational cost, $\mathcal{O}(nsl^2)$, where n is the number of generation and s the size of population at each generation.

7 Conclusions

The S^3VM is an interesting formulation of the powerful Support vector machine in a semi-supervised context. The main contribution of this paper is as follows. First, we proposed

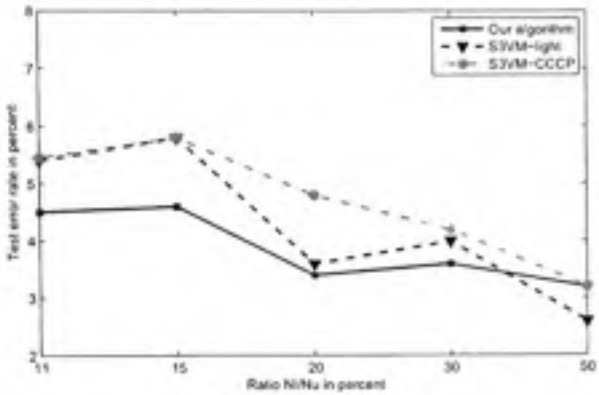


Fig. 10 Behavior of our algorithm with respect to the ratio N_l/N_u on Text dataset, where N_l represents the number of labeled samples and N_u the number of unlabeled samples.

a regularization term which improves the classifier performance. Consequently, we formulated a new objective function for S^3VM. Second, we applied GA to S^3VM and designed a special mutation operator for this problem. Third, we implemented a specific strategy based on certain heuristics to perform our proposed training algorithm by using a subset of unlabeled samples which grows at each step until convergence.

Furthermore, we tested our algorithm on artificial data as well as on real data and noted that our algorithm performs better than the standard S^3VM training algorithms. We succeeded in showing that evolutionary search contribute to find better results than classical optimization methods used for S^3VM and provide more accurate classification. Especially, when only a few labeled data is available, the search space becomes very large and other S^3VM training algorithms fall into local minima while our algorithm outperforms them.

In future work, we plan to inject more knowledge in order to reduce the computational time without loss of quality of the solution.

Acknowledgements The authors would like to thank the NSERC of Canada for their financial support.

References

1. Mathias M. Adankon and Mohamed Cheriet. Learning semi-supervised svm with genetic algorithm. In IEEE, editor, *International Joint Conference in Neural Networks 2007*, pages 1825 – 1830, orlando, FL, 2007.
2. Mathias M. Adankon and Mohamed Cheriet. Optimizing resources in model selection for support vector machines. *Pattern Recognition*, 40(3):953–963, 2007.
3. Thomas Back. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. New York, Oxford Univ. Press, 1996.
4. Thomas Back, Ulrich Hammel, and Hans paul Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1:3–17, 1997.

5. Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.
6. K. Bennett and A. Demiriz. Semi-supervised support vector machines, 1998.
7. G. Camps-Valls, T. V. Bandos Marsheva, and D. Zhou. Semi-supervised graph-based hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 45(10):3044–3054, October 2007.
8. O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.
9. O. Chapelle, V. Sindhwani, and S. S. Keerthi. Branch and bound for semi-supervised support vector machines. In *Proceedings of the NIPS 2006*, 2006.
10. O. Chapelle, V. Sindhwani, and S.S. Keerthi. Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research*, 9:203–233, 2008.
11. O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.
12. O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.
13. R. Collobert, F. Sinz, J. Weston, and L. Bottou. Large scale transductive svm. *Journal of Machine Learning Research*, 7:1687–1712, 2006.
14. A. Demiriz, K. Bennett, and M. Embrechts. Semi-supervised clustering using genetic algorithms. Technical report, R.P.I. Math Report No. 9901, Rensselaer Polytechnic Institute, Troy, New York, 1999.
15. Holger Frhlich, Bernhard Schlkopf, and Olivier Chapelle. Feature selection for support vector machines using genetic algorithms. In *International Journal on Artificial Intelligence Tools*, pages 142–148. IEEE Computer Society, 2003.
16. Frauke Friedrichs and Christian Igel. Evolutionary tuning of multiple svm parameters. In *Trends in Neurocomputing: 12th European Symposium on Artificial Neural Networks 2004*, volume 64, pages 107–117, 2005.
17. D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Massachusetts, 1989.
18. Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 529–536. MIT Press, Cambridge, MA, 2004.
19. M. Hein, J.-Y. Audibert, and U. Von Luxburg. Graph laplacians and their convergence on random neighborhood graphs. *Journal of Machine Learning Research*, 8:1325–1370, 2007.
20. J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, 1975.
21. J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, 2nd ed.* Cambridge, MA: MIT Press, 1992.
22. Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

23. Ana Carolina Lorena and André C. P. L. F. de Carvalho. Evolutionary tuning of svm parameter values in multiclass problems. *Neurocomputing*, 71(16-18):3326–3334, 2008.
24. T. Mitchell. The role of unlabeled data in supervised learning. In *Proceedings of the Sixth International Colloquium on Cognitive Science, San Sebastian, Spain.*, 1999.
25. Neil Eklund Piero P. Bonissone, Raj Subbu and Thomas R. Kiehl. Evolutionary algorithms + domain knowledge = real-world evolutionary computation. *IEEE Trans. Evolutionary Computation*, 10(3):256–280, 2006.
26. B. Scholkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
27. Matthias Seeger. Learning with labeled and unlabeled data. Technical report, Technical Report , Institute for Adaptive and neural Computation, University of Edinburgh, February 2001.
28. Marcelo M. Silva, Thiago T. Maia, and Antonio P. Braga. An evolutionary approach to transduction in support vector machines. In *Proceedings of the Fifth International Conference on Hybrid Intelligent Systems*, pages 329 – 334, 2005.
29. W. M. Spears. *The role of Mutation and Recombinaison in Evolutionary Algorithms*. Ph.D. Thesis, Fairfax, VA: George Mason University, 1998.
30. Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, volume 14, 2001.
31. V.N. Vapnik. *Statistical learning theory*. John Wiley and Sons, New York, 1998.
32. D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. 2003. In 18th Annual Conf. on Neural Information Processing Systems.
33. Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2008. http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf.

ARTICLE 2

SEMI-SUPERVISED LEAST SQUARES SUPPORT VECTOR MACHINE

M. M. Adankon[1], M. Cheriet[1], and A. Biem[2],
[1] École de Technologie Supérieure, 1100 Notre-Dame Ouest,
Montréal, Québec, Canada H3C 1K3
[2] IBM Research, Watson Research Center,
Yorktown Heights, N.Y. 10598, USA

Cet article est sous presse à IEEE Transactions on Neural Networks.

Semi-Supervised Least Squares Support Vector Machine

Mathias M. Adankon, Mohamed Cheriet and Alain Biem

Abstract

The least squares SVM, like the SVM, is based on the margin-maximization performing structural risk and has excellent power of generalization. In this paper, we consider its use in semi-supervised learning. We propose two algorithms to perform this task deduced from the transductive SVM idea. Algorithm 1 is based on combinatorial search guided by certain heuristics while Algorithm 2 iteratively builds the decision function by adding one unlabeled sample at the time. In term of complexity, Algorithm 1 is faster but Algorithm 2 yields a classifier with a better generalization capacity with only a few labeled data available. Our proposed algorithms are tested in several benchmarks and give encouraging results, confirming our approach.

Index Terms

Least squares support vector machine (LS-SVM), semi-supervised learning, transductive learning.

I. Introduction

The learning process typically assumes some form of a priori knowledge of the contextual problem at hand in the form of examplar data associated with labels. These data, called the training set, are used to design a classifier, the performance of which is measured on a separate dataset, called the testing set. This is supervised learning in which the performance of the

Mathias M. Adankon and Mohamed Cheriet are with the Synchromedia Laboratory for Multimedia Communication in Telepresence at École de Technologie Supérieure, University of Quebec, 1100 Notre-Dame West, Montreal, Canada, H3C 1K3. (email: mathias.adankon@synchromedia.ca, mohamed.cheriet@etsmtl.ca)

Alain Biem is with IBM at the Watson Research Center, New York, USA (email: biem@us.ibm.com)

Manuscript received July 3, 2008.

classifier on the test set is viewed as an estimate of the true performance of the system (i.e. the performance on the whole space). The accuracy of this approach assumes that the training set is representative of the whole space and that the data labels represent the ground truth, both conditions being necessary for a robust estimate of the true performance of the system. However, not only could the labeling process be extremely expensive and cumbersome, but human error occurring during the labeling process could lead to unpredictable results. For instance, the labeling of handwritten documents, images, or web pages requires both human expertise and insight, while in the field of medicine or biology, testing and very complex experiments are sometimes needed. Thus, it may be very difficult, or even impossible, to label all the available data.

The alternative is semi-supervised learning [27, 37], where both labeled and unlabeled data are used to train the classifier. Indeed, even without the availability of the ground truth, contextual data provide valuable information, such as the density function of the input space (if a generative parametric model is used) or the manifold structure of the data in the input space (if a structural model is used). When combined with some prior knowledge, even in succinct form, the adequate use of unlabeled data can be quite effective in improving classification performance.

Machine learning falls into two broad classes, depending on the final goal of the process: inductive learning or transductive learning. Inductive learning pursues the standard goal in machine learning, which is to accurately classify the entire input space. In contrast, transductive learning focuses on a predefined target set of unlabeled data, the goal in this case being to label the specific target set, an apparently simpler problem than that of induction. Support vector machines (SVMs) have gained a great deal of attention as powerful classification systems. They have a solid theoretical foundation, rooted in statistical learning theory, and constitute a non-generative approach which maps data into a high-dimensional space in which linear classification can be performed while maximizing the margin of the separating hyperplane. The transductive SVM (TSVM) introduced by Vapnik [34] was originally designed to directly estimate labels for unlabeled samples (the target test set). TSVM maximizes the margin hyperplane using both labeled and unlabeled data, and implements the cluster assumption for semi-supervised learning [13]. Thus, TSVM learning provides a decision boundary in the entire input space and can be

considered as inductive, rather than strictly transductive, semi-supervised learning as reported by [12] *"This idea was first introduced by Vapnik and Sterin (1977) under the name Transductive SVM, but since it learns an inductive rule defined over the entire input space, we refer to this approach as Semi-Supervised SVM (S^3VM)"*.

The classifier is designed to correctly classify objects unseen during the training process. Generalization represents the capacity to classify these unseen data. Thus, an estimate of the generalization capacity of the classifier is done on a representative set called the test set, not seen during training. However, the notion of test set depends on the underlying learning philosophy. For transductive learning, the test set is a predetermined set of unlabeled data for which labels ought to be estimated; the unlabeled data are used in the learning process to estimate the unknown labels. In contrast, semi-supervised learning uses an unlabeled set as extra information for the goal of classifying the entire space and a separate test set is used as a representative sample of that space for accuracy estimation.

The semi-supervised SVM (S^3VM) can considerably improve the generalization potential of SVMs when the ratio of the number of labeled points to the number of unlabeled points is small. However, a larger number of examples yields a more difficult optimization problem, because of the non-convex nature of the objective function. Various solutions have been proposed with varying degrees of success, including an integer programming method [4], a combinatorial approach [23], and a sequential optimization procedure [18] with good scalable properties, albeit only demonstrated in the linear case. An interesting approach is given in [5] where the non-convex transductive problem is transformed into a convex, semi-definite programming problem. In general, the optimization problem is simplified with an appropriate choice of loss function. Further details on recent S^3VM optimization techniques can be found in [12].

Least squares SVMs (LS-SVMs) have been proposed as a way to replace the convex quadratic programming problem with a convex linear system solving problem [30]. This is achieved by changing the hinge loss function by a squared-loss function. It has been shown through a meticulous empirical study that the generalization performance of the LS-SVM is comparable to that of the SVM [33]. In addition, the training algorithm of the LS-SVM is greatly simplified,

since a linear problem is resolved instead of a quadratic programming (QP) problem in the SVM case.

Various algorithms are available for supervised LS-SVM to deal with training [30, 15, 14], model selection [10, 8, 2] and sparseness [29, 9, 17, 22]. However, to our knowledge, only one work about the semi-supervised LS-SVM is available. In this work, Luts et al. [25] introduced an extra variable and additional equality constraints into the classifier formulation in order to avoid the zero label assumption.

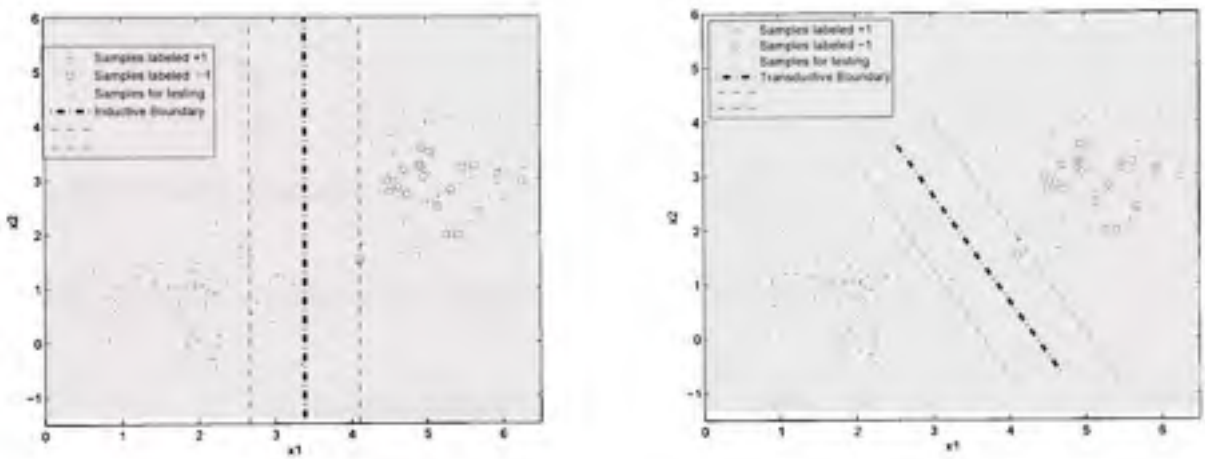
While the LS-SVM has proven to be a successful alternative to standard SVMs, the number of applications in semi-supervised settings has been limited, due in large part to the optimization problem outlined earlier when a large unlabeled dataset is used. The goal of this paper is to provide a solution to this problem by proposing two learning methods applicable to semi-supervised learning. In particular, we introduce an LS-SVM optimization framework for semi-supervised learning using the transductive idea from the TSVM approach of Vapnik [35], and develop two training algorithms in this framework. The first algorithm labeled Algorithm 1, based on combinatorial search guided by certain heuristics, is inspired by the Joachims' approach used to learn the TSVM [23]. The second algorithm, Algorithm 2, iteratively builds the decision function by labeling one unlabeled sample at a time. The two algorithms provide a good performance. However, when only a few labeled data are available, Algorithm 2 performs better with a high computational cost.

This paper is organized as follows: In section 2, we give a review of the TSVM. In section 3, we present the semi-supervised LS-SVM ($S^2LS-SVM$) deduced from the TSVM. In section 4, we describe the two methods developed for training the $S^2LS-SVM$. In section 5, we present the experimental results on both artificial and real data. In the last section, we conclude the paper.

II. Transductive Support Vector Machine

This section describes the TSVM as developed in [35] by Vapnik. Classically, the use of inductive inference for estimating the value of a function at given points involves two steps. First, the training points are used to estimate a function for the entire input space, and second,

the values of that function on separate test points are computed based on the estimated parameters. In contrast, in transductive inference, the end goal is to determine the values of the function at the predetermined test points, which, as pointed by [35] is a simpler problem than the inductive problem. Indeed, in [34], it is demonstrated that, as long as the end goal is the value of the function at given selected points, the transductive approach is more accurate and direct than the inductive approach, and yields better results, since the use of the test set can be reframed as part of the learning problem. Figure 1 shows a comparison of inductive and transductive learning, where the classification boundary is shown for the inductive SVM and the transductive SVM.



(a) Boundary found by Inductive learning with SVM classifier. Only labeled points (+ and o) are used for training.

(b) Boundary found by Transductive learning with SVM classifier. All points, labeled and testing(unlabeled), are used for training. Good classification was obtained on the testing samples.

Fig. 1. Illustration of the Transductive accuracy in a two-dimensional input space.

The TSVM can be described as follows. We consider a binary classification problem with training data $\{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}$ and test data $\{x_1^\star, \ldots, x_n^\star\}$, $x_i \in \mathbb{R}^d$, $x_i^\star \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$.

In inductive learning, the training algorithm of the SVM resolves implicitly the following optimization problem, which expresses the maximization of the margin $2/\|w\|$ and the minimization of the training error:

$$\min_{w,b,\xi} \frac{1}{2} w'w + C \sum_{i=1}^{\ell} \xi_i \tag{1}$$

$$\text{s.t.} \quad y_i[w'\phi(x_i) + b] \geq 1 - \xi_i \quad \forall i = 1, ..., \ell \tag{2}$$

$$\xi_i \geq 0 \quad \forall i = 1, ..., \ell \tag{3}$$

where w' denotes the transpose of w, $< w, b >$ is the model parameter, C is used to balance the trade-off between maximizing the margin and minimizing the training error and ξ is the slack variable that quantifies SVM training error.

In the primal form, the Lagrangian of this problem is as follows:

$$\mathcal{L} = \frac{1}{2} w'w + C \sum_{i=1}^{\ell} \xi_i - \sum_{i=1}^{\ell} \alpha_i [y_i (w'\phi(x_i) + b) - 1 + \xi_i] - \sum_{i=1}^{\ell} \lambda_i \xi_i \tag{4}$$

with the Lagrange multipliers $\alpha_i \geq 0$ and $\lambda_i \geq 0$ for all $i = 1, ..., \ell$.

After taking the conditions for optimality, we obtain the classifier:

$$f(x) = sign[w'\phi(x) + b] = sign[\sum_{i=1}^{\ell} \alpha_i y_i k(x_i, x) + b] \tag{5}$$

with α the solution of :

$$max \quad W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{6}$$

$$s.t. \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0 \quad and \quad 0 \leq \alpha_i \leq C, i = 1, ..., \ell$$

where $k(x_i, x_j) = \phi(x_i)'\phi(x_j)$ is the kernel function.

The threshold b is computed by averaging $b = y_j - \sum y_i \alpha_i k(x_i, x_j)$ over all support vectors x_j ($\alpha_j > 0$).

The TSVM is an interesting version of the SVM in which transductive inference is used. In this case, the TSVM attempts to find the hyperplane $< w, b >$ and the labels $y_1^*, \ldots, y_n^*$ of the

test data that maximize the margin with minimum error, thereby obtaining the label of the test data in one step:

$$\min_{w,b,\xi,\xi^*,y_1^*,\ldots,y_n^*} \mathcal{J} = \frac{1}{2}w'w + C\sum_{i=1}^{\ell}\xi_i + C^*\sum_{j=1}^{n}\xi_j^* \tag{7}$$

$$\text{s.t.} \quad y_i[w'\phi(x_i)+b] \geq 1-\xi_i, \quad \xi_i \geq 0 \quad \forall i = 1,\ldots,\ell \tag{8}$$

$$y_j^*[w'\phi(x_j^*)+b] \geq 1-\xi_j^*, \quad \xi_j^* \geq 0 \quad \forall j = 1,\ldots,n \tag{9}$$

$$y_j^* \in \{-1,1\} \quad \forall j = 1,\ldots,n \tag{10}$$

Vapnik [35] proposed this formulation to reinforce the classifier accuracy on the specific test set by adding the minimization of the error on the test set $\{x_1^*,\ldots,x_n^*\}$. We note that equation (7) minimizes both the training error and the test error. Solving the problem expressed by equation (7) is then equivalent to finding the hyperplane that separates both the training data and the test data with minimum cost error.

Using the combinatorial optimization approach, the goal is first to find the set of binary variables. Then, for the fixed $y_1^*,\ldots,y_n^*$ we obtain the following functional, which must be maximized:

$$\begin{aligned} W_{y_1^*,\ldots,y_n^*}(\alpha,\alpha^*) = &\sum_{i=1}^{\ell}\alpha_i - \frac{1}{2}\sum_{i,r=1}^{\ell}\alpha_i\alpha_r y_i y_r k(x_i,x_r) \\ &+ \sum_{j=1}^{n}\alpha_j^* - \frac{1}{2}\sum_{j,r=1}^{n}\alpha_j^*\alpha_r^* y_j^* y_r^* k(x_j^*,x_r^*) \\ &- \sum_{i=1}^{\ell}\sum_{j=1}^{n}\alpha_i\alpha_j^* y_i y_j^* k(x_i,x_j^*) \end{aligned}$$

$$\begin{aligned} s.t. \quad & 0 \leq \alpha_i \leq C, i = 1,\ldots,\ell \\ & 0 \leq \alpha_j^* \leq C^*, j = 1,\ldots,n \\ & \sum_{i=1}^{\ell}\alpha_i y_i + \sum_{i=j}^{n}\alpha_j^* y_j^* = 0. \end{aligned} \tag{11}$$

Optimizing the problem of equation (7) with respect to all $(y_1^*, \ldots, y_n^*)$ is combinatorial problem whose size increase with n. Finding the exact solution requires searching over all possible 2^n labels of the unlabeled data. Clearly, this is not feasible with a large number of data.

In 1999, Joachims [23] proposed a combinatorial approach to solving the TSVM problem with certain heuristics, the implementation of which solves large problems; however, convergence is still not guaranteed. Recently, Chapelle et al. [11] proposed applying branch-and-bound techniques to obtain a globally optimal solution. While their method gives a good solution, its implementation does not scale to a large dataset. Interestingly, the authors conclude that transductive inference can be used to formulate semi-supervised learning with promising results by using unlabeled data in the place of test data. Indeed, maximizing the margin under both labeled and unlabeled data provides a decision boundary which lies in a low-density region (cluster assumption for semi-supervised learning) [13].

III. Semi-supervised LS-SVM formulation ($S^2LS-SVM$)

The LS-SVM is an interesting variant of the SVM proposed by Suykens et al.[31, 30]. The standard Vapnik SVM classifier is modified to transform the QP problem into a linear one. These modifications are formulated in the LS-SVM definition as follows:

$$\min_{w,b,\xi} \frac{1}{2} w'w + \frac{1}{2} C \sum_{i=1}^{\ell} \xi_i^2 \tag{12}$$

$$\text{s.t.} \quad \xi_i = y_i - \left[w'\phi(x_i) + b\right] \quad \forall i = 1, ..., \ell \tag{13}$$

The original SVM formulation is modified at two points. First, the inequality constraints with the slack variable ξ_i expressed in (2) are replaced by equality constraints. Second, a squared loss function is considered in the objective function. These two essential modifications simplify the problem, which becomes linear.

The Lagrangian of problem (12) is expressed by :

$$\mathcal{L}(w,b,\xi,\alpha) = \frac{1}{2}w'w$$
$$+\frac{1}{2}C\sum_{i=1}^{\ell}\xi_i^2 + \sum_{i=1}^{\ell}\alpha_i\{y_i - [w'\phi(x_i)+b] - \xi_i\}$$

where α_i are Lagrange multipliers.

The Karush−Kuhn−Tucker (KKT)[1] conditions for optimality yield

$$\begin{cases} \frac{\partial\mathcal{L}}{\partial w} = 0 & \Rightarrow \quad w = \sum_{i=1}^{\ell}\alpha_i\phi(x_i) \\ \frac{\partial\mathcal{L}}{\partial b} = 0 & \Rightarrow \quad \sum_{i=1}^{\ell}\alpha_i = 0 \\ \frac{\partial\mathcal{L}}{\partial\xi_i} = 0 & \Rightarrow \quad \alpha_i = C\xi_i, \quad \forall i = 1,...,\ell \\ \frac{\partial\mathcal{L}}{\partial\alpha_i} = 0 & \Rightarrow \quad \xi_i = y_i - [w'\phi(x_i)+b] \quad \forall i = 1,...,\ell \end{cases}$$

We note that the system coming from the KKT conditions is linear, and that its solution is found by solving the system of linear equations expressed in the following matrix form :

$$\begin{pmatrix} K + C^{-1}I & \vec{1}' \\ \vec{1} & 0 \end{pmatrix}\begin{pmatrix} \alpha \\ b \end{pmatrix} = \begin{pmatrix} Y \\ 0 \end{pmatrix} \tag{14}$$

where :

$K_{ij} = k(x_i, x_j)$

$Y = (y_1,...,y_\ell)'$

$\alpha = (\alpha_1,...,\alpha_\ell)'$

$\vec{1} = (1,...,1)$

With regard to equation (7), we can formulate the $S^2LS-SVM$ using the following expressions:

[1]KKT conditions are necessary conditions for optimality obtained from first derivative.

$$\min_{w,b,\xi,\xi^*,y_1^*,\ldots,y_n^*} \frac{1}{2}w'w + \frac{1}{2}C\sum_{i=1}^{\ell}\xi_i^2 + \frac{1}{2}C^*\sum_{j=1}^{n}(\xi_j^*)^2 \tag{15}$$

$$\text{s.t.} \quad \xi_i = y_i - [w'\phi(x_i) + b], \quad \forall i = 1, ..., \ell \tag{16}$$

$$\xi_j = y_j^* - [w'\phi(x_j^*) + b], \quad \forall j = 1, ..., n \tag{17}$$

$$y_j^* \in \{-1, 1\} \quad \forall j = 1, ..., n \tag{18}$$

In this equation, the parameters C and C^* balance the error between the labeled and unlabeled data.

Considering the combinatorial view of the optimization problem (15), the variables w, b, ξ, ξ^* and y^* are optimized at different levels. Then, for a given fixed set $y_1^*, \ldots, y_n^*$, the optimization over (w, b) is standard LS-SVM training, and we obtain a linear system in dual space expressed in matrix form by:

$$\begin{pmatrix} K + \Gamma & \vec{1}' \\ \vec{1} & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ b \end{pmatrix} = \begin{pmatrix} Y \\ 0 \end{pmatrix} \tag{19}$$

where:

$K_{ij} = k(x_i, x_j)$

$Y = (y_1, ..., y_\ell, y_1^*, ..., y_n^*)'$

$\alpha = (\alpha_1, ..., \alpha_\ell, \alpha_1^*, ..., \alpha_n^*)'$

$\vec{1} = (1, ..., 1)$

Γ is diagonal matrix with $\Gamma_{ii} = 1/C$ for $i = 1, ..., \ell$ and $\Gamma_{ii} = 1/C^*$ for $i = \ell + 1, ..., \ell + n$

IV. Learning Algorithms

In this section, we describe the two algorithms we propose for solving the semi-supervised problem expressed in equation (15).

A. Algorithm 1

Our first approach is based on the strategy developed by Joachims in learning the TSVM [23]. We use a combinatorial search approach guided by certain heuristics. We start by learning with labeled data and we use the resulting classifier to label the unlabeled data constrained by the ratio of positive and negative samples. In addition, during the optimization process, we use heuristics to change the labels of the unlabeled samples with the largest corresponding errors ξ_j^*. Algorithm 1 illustrates the details of the strategies we used.

Algorithm 1 Training semi-supervised LS-SVM

Input Labeled samples $\{X, Y\}$ and Unlabeled samples X^*

Output Hyperplane $< w, b >$

Train the machine with labeled samples

Compute the output of each unlabeled samples

Label them by using the defined ratio

while Objective function decreases **do**

  Train the machine with all samples

  Select the unlabeled sample whose corresponding $|\alpha^*|$ is the largest

  if $y^*\alpha^* > 0$ **then**

    Change the label into its opposite

  end if

end while

Considering the equation $\alpha_j^* = C^*\xi_j^*$, it is clear that the error is proportional to the value of α^*. So, the objective function can be written as follows:

$$\mathcal{G} = \frac{1}{2}\|w\|^2 + \frac{1}{2C}\sum_i(\alpha_i)^2 + \frac{1}{2C^*}\sum_j(\alpha_j^*)^2 \tag{20}$$

Thus, we use the value of α_j^* as a criterion for selecting the unlabeled sample with the largest error, and change the corresponding label into its opposite. However, only the labels of samples misclassified according to the margins are changed.

If a sample x_j^* is misclassified according to the margin, we have :

$$y_j^* f(x_j^*) \leq 1$$

which can been written as:

$$y_j^* [y_j^* - f(x_j^*)] \geq 0 \tag{21}$$

Then, without computing $f(x_j^*)$, it possible to identify the unlabeled sample misclassified according to the margin by testing the following inequality:

$$y_j^* \alpha_j^* \geq 0 \tag{22}$$

The objective function is reduced at each step of the algorithm, under the constraint of keeping a similar ratio of positive versus negative examples between labeled and unlabeled data. In our case, we use the ratio of the positive samples in the unlabeled set defined by :

$$R = \frac{1}{2}\left[1 + \frac{1}{n}\sum_{j=1}^{n} y_j^*\right] \tag{23}$$

In general, when the labeled set is too small, we need to use this balancing constraint in order to avoid unbalanced solutions. Without that constraint, the algorithm assigns all unlabeled samples in the same class when $\ell \ll n$.

Tracking this ratio is the main disadvantage of this method, as the ratio is important to enforcing a balanced solution. The ratio can be viewed as a way to include a priori knowledge in the data distribution derived from the labeled data. Performance may decrease significantly when the estimated ratio deviates from the actual estimate. More troublesome is the fact that balancing the ratio between labeled and unlabeled sets may not be the correct approach. Unlabeled data may exhibit a different distribution of labeled data versus unlabeled data, in particular for larger unlabeled datasets. Relying too much on this ratio is, therefore, a complex and risky approach.

B. Algorithm 2

To overcome the sensitivities of the ratio parameter, we propose an alternative to Algorithm 1. In this alternative scheme, the ratio is not specified a priori. Instead, unlabeled examples are labeled gradually during the learning process: one sample is labeled and added to the labeled set.

The added sample is chosen in order to obtain the smallest increase in the objective function. The criterion we use to select this point is based on the value of α_j^*.

First, we identify the label of the point to be labeled according to the objective function. Next, for each remaining unlabeled sample, we compute $a_j^{(1)} = \alpha_j^*$ if the identified label is 1 and $a_j^{(-1)}$ for the opposite label. As the goal is to find the unlabeled sample x^* with the smallest increase in the objective function, we select, at each step, the unlabeled sample, the corresponding α_j^* of which will be the smallest if it is added to the previous solution. The details of this second approach are shown in the Algorithm 2.

Algorithm 2 Progressive Learning for semi-supervised LS-SVM

Input Labeled samples $\{X, Y\}$ and Unlabeled samples X^*

Output Hyperplane $< w, b >$

Train the machine with labeled samples

while Unlabeled sample remains **do**

Select the sign of the label according to the objective function

Compute $a_j^{(1)}$ or $a_j^{(-1)}$ for each remaining unlabeled sample

if label is positive **then**

Find the sample whose corresponding $a_j^{(1)}$ is the minimum and Label it 1

else

Find the sample whose corresponding $a_j^{(-1)}$ is the minimum and Label it -1

end if

end while

Re-train the final machine

C. An incremental version of the Algorithm 2

In this section, we propose an incremental version of Algorithm 2, and propose an algorithm that compute α_j^* for a new sample using the previous solution. For incremental supervised learning, several methods have been proposed elsewhere in the framework of SVM and LS-SVM [24, 7, 19]. The focus here is on semi-supervised learning.

We suppose that we have the solution of the previous machine (α^0, b^0), and we want to compute the value α_j^* of the added point x_j^*. Using eq. (19) and the decomposition matrix, we have :

$$\begin{pmatrix} \alpha_j^* \\ \alpha \\ b \end{pmatrix} = \begin{pmatrix} K_{jj} + 1/C^* & K_j & 1 \\ K_j' & K + \Gamma & \vec{1}' \\ 1 & \vec{1} & 0 \end{pmatrix}^{-1} \begin{pmatrix} y_j^* \\ Y \\ 0 \end{pmatrix} \tag{24}$$

Applying the block matrix inversion lemma [28], we obtain :

$$\alpha_j^* = \begin{pmatrix} \frac{1}{\beta} & -\frac{1}{\beta} \begin{pmatrix} K_j & 1 \end{pmatrix} \begin{pmatrix} K + \Gamma & \vec{1}' \\ \vec{1} & 0 \end{pmatrix}^{-1} \end{pmatrix} \begin{pmatrix} y_j^* \\ Y \\ 0 \end{pmatrix} \tag{25}$$

with

$$\beta = K_{jj} + 1/C^* - \begin{pmatrix} K_j & 1 \end{pmatrix} \begin{pmatrix} K + \Gamma & \vec{1}' \\ \vec{1} & 0 \end{pmatrix}^{-1} \begin{pmatrix} K_j \\ 1 \end{pmatrix} \tag{26}$$

After some manipulations, the following result is obtained :

$$\alpha_j^* = \frac{1}{\beta}[y_j^* - f^0(x_j^*)] \tag{27}$$

where $f^0(x_j^*)$ is the output of the previous machine (α^0, b^0) for the sample x_j^* and

$$\begin{pmatrix} \alpha^0 \\ b^0 \end{pmatrix} = \begin{pmatrix} K + \Gamma & \vec{1}' \\ \vec{1} & 0 \end{pmatrix}^{-1} \begin{pmatrix} Y \\ 0 \end{pmatrix}.$$

Then, from equation (27), we have :

$$a_j^{(1)} = \frac{1}{\beta}[1 - f^0(x_j^*)] \tag{28}$$

and

$$a_j^{(-1)} = \frac{1}{\beta}[-1 - f^0(x_j^*)] \tag{29}$$

Algorithm 2 is motivated by the failure of Algorithm 1 on the **Two Moons** dataset which is a standard benchmark for the semi-supervised learning algorithms used in the literature [21, 6, 3, 11, 36]. This problem is difficult for most semi-supervised SVM algorithms [11], but, with our second algorithm, based on the progressive labeling process, it is easy to construct the right boundary. In terms of complexity, Algorithm 1 is faster. Assuming that the complexity of the LS-SVM training algorithm is $\mathcal{O}(n^3)$[2] , if we need r iterations to perform the optimization problem, the total computational cost of Algorithm 1 is $\mathcal{O}(rn^3)$. For Algorithm 2, the computational cost becomes $\mathcal{O}(n^4)$.

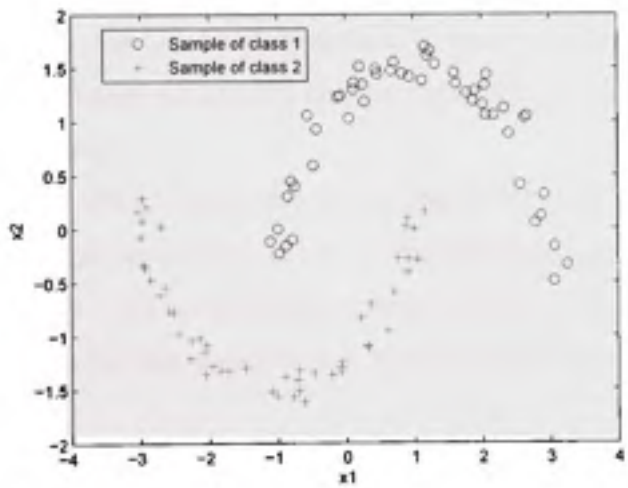


Fig. 2. Illustration of the Two Moons problem

V. Experiments

A. Data Sets

1) Gaus50: We consider a two-class problem in a 50-dimensional input space like the artificial datasets used in [20]. The data of each class are generated with equal probability from a Gaussian

[2]Basic training algorithm is $\mathcal{O}(n^3)$ but the complexity is improved with other techniques, e.g. conjugate gradient.

distribution with a unit covariance matrix. The mean of the Gaussian is $(a, a, ..., a)$ for class 1 and $-(a, a, ..., a)$ for the class 2. In order to obtain a Bayes error equal to 5%, the parameter a is fixed to 0.23; but with supervised LS-SVM, we get 5.7% error rate on the test set. We generated 550 samples for training and 1000 samples for testing.

2) Gaus50x: We have here another artificial two-class problem in a 50-D space but with multimodal distribution. The data of each class are generated with equal probability from a Gaussian mixture distribution with significant overlap. The two classes have equal prior and their class-conditional distributions are respectively $p(x|y = 1) = 0.49\mathcal{N}(\mu_1, I) + 0.51\mathcal{N}(\mu_2, I)$ and $p(x|y = -1) = 0.49\mathcal{N}(-\mu_1, I) + 0.51\mathcal{N}(-\mu_2, I)$ where $\mu_1 = (0.25, 0.25, ..., 0.25, 0.25)$, $\mu_2 = (0.25, 0.25, ..., -0.25, -0.25)$, $N(\mu, I)$ is a Gaussian density function with its mean vector μ, and I is the identity matrix for its covariance matrix. The supervised LS-SVM gives a 12.4% error rate on the test set. We generated 550 samples for training and 1000 samples for testing.

3) Text: Text document classification is a problem involving high dimensionality. We used the *mac* and *windows* classes taken from the 20 news group dataset, with 7511 dimensions, as preprocessed in [32]. We divided the dataset in two subsets : 1446 samples for training and 500 for testing.

4) USPS: USPS is the well known US Postal Service handwritten digits recognition corpus. The digits are represented by normalized grey scale images of size 16×16. The learning dataset contains 7291 samples for training while the testing dataset consists of 2007 other samples. We have multi-class problem with 10 classes. We then trained 10 machines using the *one-against-all* strategy.

B. Comparison between the supervised and semi-supervised LS-SVMs

To quantify the effect of semi-supervised learning, we ran a set of experiments using the LS-SVM classifier trained with either labeled data or a mix of labeled and unlabeled data. We randomly selected 5% of the training set to form the labeled data, the remaining data consisting of the unlabeled samples. We repeated this operation 10 times, each time testing our two semi-supervised algorithms and the single LS-SVM trained only with labeled data. The results are shown in Tables I, II and III. We used the RBF kernel and the kernel parameter $\gamma = 1/\sigma^2 = 0.02$. The hyperparameters C and C^* are set to 1 and 0.1 respectively for the data "Gaus50", $C = 1$

and $C^* = 10$ for "Gaus50x" and $C = C^* = 1$ for the Text documents problem.

TABLE I
ERROR RATE (%) ON THE DATA SET "GAUS50"

Selection	LS-SVM with labeled data	Semi-supervised LS-SVM Algorithm 1	Semi-supervised LS-SVM Algorithm 2
#1	11.10	6.10	6.10
#2	32.50	8.50	6.60
#3	22.70	5.30	5.40
#4	44.80	7.60	6.10
#5	44.60	9.10	6.10
#6	48.50	6.90	5.60
#7	12.30	6.70	5.90
#8	11.50	5.50	5.80
#9	14.90	6.30	6.20
#10	13.00	6.40	5.70
Mean	25.84	6.88	5.95

We have two main comments on the results reported in Tables I, II and III. First, semi-supervised learning gives good results. Adding unlabeled data improves the capacity of generalization of the classifier, and sometimes we approach the Bayes error. For example, the Bayes error for the artificial problem is 5% and the semi-supervised Algorithm 1 gives a 5.3% error rate for selection #3, while Algorithm 2 gives a 5.4% error rate. Also, we note that the two semi-supervised algorithms are robust, even though the distribution information given by the labeled samples differ from the true distribution, for selections #4, #5, and #6 in Table I, #2, #4, #5 and #6 in Table II and all the selections in Table III. Second, the results in Tables I, II and III show that Algorithm 2 performs, on average, better than Algorithm 1, confirming the advantage of the incremental labeling approach. These latest results justify our decision to relax the explicit constraint of the ratio between positive and negative examples in the unlabeled data.

C. Comparison of our semi-supervised LS-SVMs with the semi-supervised SVM

In this section, we compare our two algorithms with the S^3VM. The goal is to test the robustness of our proposed LS-SVM-based algorithm relative to the classical SVM-based alternatives.

TABLE II

ERROR RATE (%) ON THE DATA SET "GAUS50X"

Selection	LS-SVM with labeled data	Semi-supervised LS-SVM Algorithm 1	Semi-supervised LS-SVM Algorithm 2
#1	18.10	13.60	13.20
#2	47.70	20.50	15.40
#3	34.60	16.90	14.20
#4	47.50	17.40	12.50
#5	48.80	19.10	14.90
#6	49.90	19.20	14.60
#7	20.10	17.00	13.90
#8	24.90	15.80	13.80
#9	19.70	13.40	13.50
#10	27.00	19.70	14.40
Mean	33.86	17.26	14.04

TABLE III

ERROR RATE (%) ON THE DATA SET "TEXT"

Selection	LS-SVM with labeled data	Semi-supervised LS-SVM Algorithm 1	Semi-supervised LS-SVM Algorithm 2
# 1	48.80	4.80	5.60
# 2	50.40	4.60	3.60
# 3	50.40	4.40	3.20
# 4	50.20	3.30	5.00
# 5	50.20	5.80	5.80
# 6	49.60	5.80	5.60
# 7	49.60	5.40	4.80
# 8	50.40	4.80	4.20
# 9	50.20	4.00	3.80
# 10	49.60	4.60	5.20
Mean	49.94	4.76	4.68

Indeed, both our proposed algorithms resolve the semi-supervised LS-SVM problem through a transductive formulation, which calls for a comparison with the S^3VM algorithms using the idea of transduction. Here, the S^3VM is different in terms of the loss function and the optimization technique used. The following S^3VM algorithms were tested:

- $S^3VM - light$ [23], which uses a combinatorial search over the unlabeled data while keeping the ratio of positive versus negative the same as the ratio in the labeled set;
- $S^3VM - GA$, which uses the Genetic Algorithm to find the optimal solution over the labeled and unlabeled dataset [1];
- $S^3VM - CCCP$ (concave-convex procedure), which decomposes a non-convex loss function into a convex part and a concave part, and where this last part is iteratively approximated by a Taylor expansion [16].

We used all the four datasets described in section 4.1 and the same experimental setup as described in [1]. For each dataset, we labeled 10% of the training samples and kept the rest as unlabeled data. The error rate is computed on the test set. The complete experimental setup is summarized in Table IV.

The results are reported in Table V. We note that semi-supervised LS-SVM achieved a good performance and its generalization performance is comparable to that of semi-supervised SVM.

TABLE IV

EXPERIMENTAL SETUP, n_l AND n_u CORRESPONDS RESPECTIVELY TO THE NUMBER OF LABELED SAMPLES AND UNLABELED SAMPLES. WE KEEP THE SAME SETUP IN THE FOLLOWING SECTIONS.

	γ	C	C^*	n_l	n_u
Gaus50	0.02	1	0.1	55	495
Gaus50x	0.02	1	10	55	495
Text	0.02	1	1	144	1302
USPS	0.004	0.018	0.018	729	6562

TABLE V

COMPARISON BETWEEN SEMI-SUPERVISED SVM AND SEMI-SUPERVISED LS-SVM: $S^3VM - light$ [23], $S^3VM - CCCP$ [16] AND $S^3VM - GA$ [1] LEARN SEMI-SUPERVISED SVM WHILE $S^2LS - SVM1$ AND $S^2LS - SVM2$ LEARN SEMI-SUPERVISED LS-SVM. IN THE TABLE WE REPORT ERROR RATE ON TEST SET.

	$S^3VM - light$	$S^3VM - CCCP$	$S^3VM - GA$	$S^2LS - SVM1$	$S^2LS - SVM2$
Gaus50	8.80%	7.70%	5.70%	6.10%	**5.50%**
Gaus50x	16.30%	15.50%	14.70%	16.50%	**14.00%**
Text	5.40%	5.45%	**4.00%**	5.00%	4.20%
USPS	14.70%	7.32%	7.62%	10.31%	**6.72%**

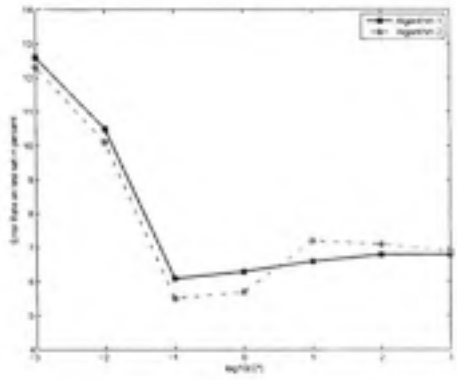
D. *Impact of hyperparameters*

To train semi-supervised LS-SVM, the hyperparameters (kernel, C and C^*) need to be set to their optimal values. We know that the choice of kernel corresponds to the selection of a learning function space. In fact, the kernel determines the functional form of all the possible solutions. Thus, the choice of kernel is very important to building a good classifier [26].

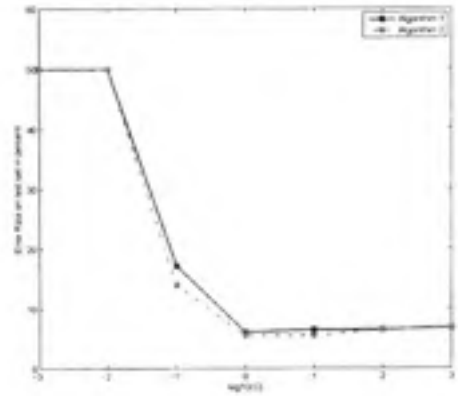
In this section, we focus on the hyperparameters C and C^*, which :
(i) balance the error between the labeled and unlabeled data and,
(ii) control the trade-off between maximizing the margin and minimizing the total cost error on the training data.

We perform various experiments to show the influence of these hyperparameters on the capacity of the generalization of the classifier built in semi-supervised context. In this section, we begin the experiments with the *Gaus50* dataset.

First, we set C equal to 1. Then, using various values of C^*, we vary the ratio of the error between the labeled and unlabeled data. Second, we made C and C^* vary, while keeping the ratio C/C^* equal to 10. This experiment is performed to show that it is not sufficient to balance the error between the labeled and unlabeled data, but it is also important to balance the trade-off between maximizing the margin and minimizing the total cost error.



(a) Impact of the ratio C/C^* in semi-supervised LS-SVM, we set $C = 1$ and vary C^*.



(b) Impact of the value of C and $C*$ on the classifier if the ratio C/C^* is fixed, varying C and $C*$ with constraint $C/C^* = 10$.

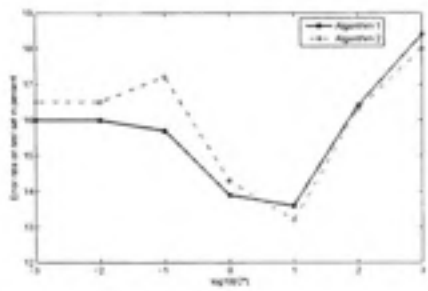
Fig. 3. Impact of the hyperparameters on the "Gaus50" dataset.

The results obtained, shown in Figures 3(a) and 3(b), reveal that it is important to choose an optimal value for both C and C^* in order to achieve a good performance. We also note that both algorithms proposed for training semi-supervised LS-SVM, behave in the same way with respect to the hyperparameters.

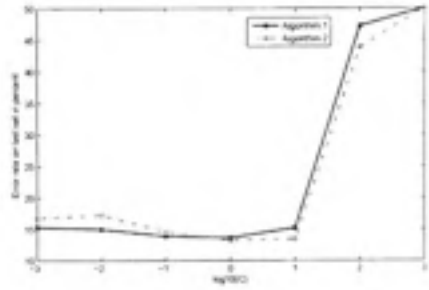
We repeated the same experiment with the "Gaus50x" and "Text" datasets and obtained the same results, as shown in Figures 4 and 5.

E. Impact of the ratio of the labeled samples

We also tested the behavior of our algorithm with respect to the ratio N_l/N_u, where N_u represents the number of unlabeled samples and N_l the number of labeled samples. Figures 6, 7 and 8 show the variation of the test error when we increased the number of labeled samples in the training data. We see that the test error for Algorithm 1 drops as the ratio N_l/N_u increases. Since we have more information about the samples labels, the generalization error is improved.

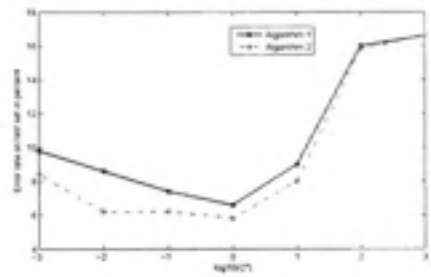


(a) Impact of the ratio C/C^* in semi-supervised LS-SVM, we set $C = 1$ and vary C^*.

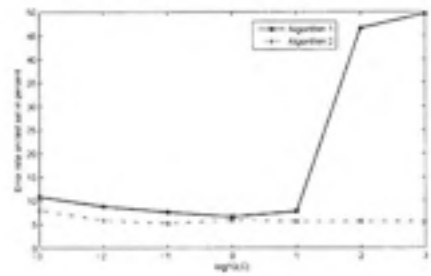


(b) Impact of the value of C and $C*$ on the classifier if the ratio C/C^* is fixed, varying C and $C*$ with constraint $C/C^* = 0.1$.

Fig. 4. Impact of the hyperparameters on the "Gaus50x" dataset.



(a) Impact of the ratio C/C^* in semi-supervised LS-SVM, we set $C = 1$ and vary C^*.



(b) Impact of the value of C and $C*$ on the classifier if the ratio C/C^* is fixed, varying C and $C*$ with constraint $C/C^* = 1$.

Fig. 5. Impact of the hyperparameters on the "Text" dataset.

However, we note that the test error rate obtained with Algorithm 2 is good and independent of the ratio N_l/N_u. This is in line with the rationale for Algorithm 2, which is to not rely on the ratio of positive to negative examples. Furthermore, the classifier built with Algorithm 2 has a good generalization capacity with a few labeled samples, suggesting that it has the ability to infer the internal structure of the data in a more effective way.

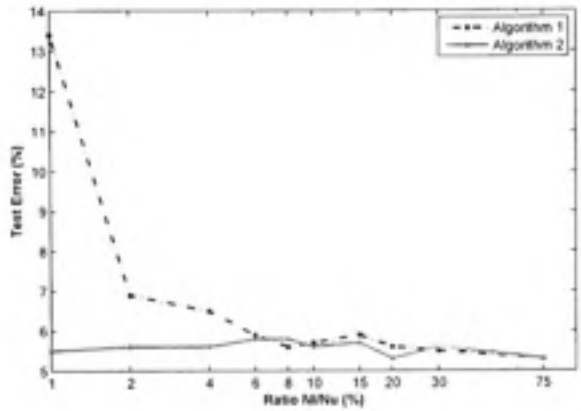


Fig. 6. "Gaus50" dataset: Test error w.r.t. ratio N_l/N_u, where N_u represents the number of unlabeled samples and N_l the number of labeled samples.

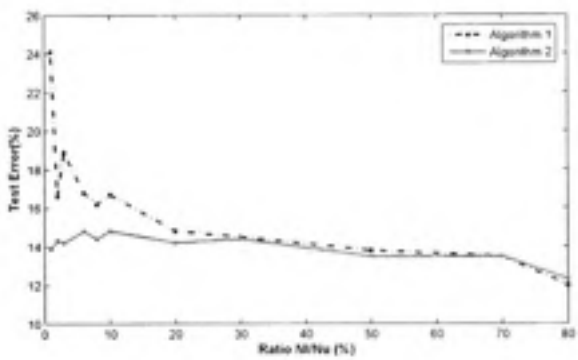


Fig. 7. "Gaus50x" dataset:Test error w.r.t. ratio N_l/N_u, where N_u represents the number of unlabeled samples and N_l the number of labeled samples.

F. Effect of an error in the unlabeled samples on classifier accuracy

In this section, we tested the influence of an error in the unlabeled samples on classifier accuracy in order to find the relation (correlation) between the error rate on the unlabeled samples and the generalization error (test error). We use the various semi-supervised problems generated by changing the selected labeled samples. For each selection, after training semi-supervised LS-SVM with Algorithm 1 and Algorithm 2, we compute the error rate on the unlabeled samples (because we know the true labels) and on the test set. The results obtained with Algorithm 1 are plotted in Figures 9, 10 and 11 while those for Algorithm 2 are shown in Figures 12, 13

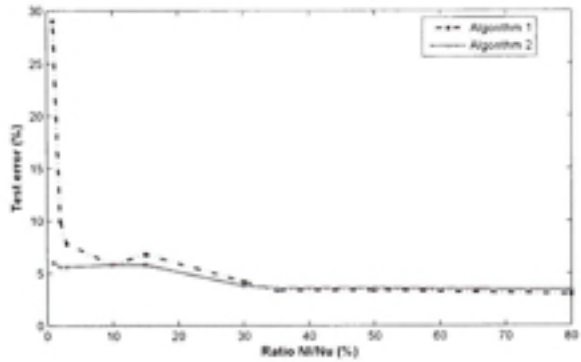


Fig. 8. "Text" dataset:Test error w.r.t. ratio N_l/N_u, where N_u represents the number of unlabeled samples and N_l the number of labeled samples.

and 14.

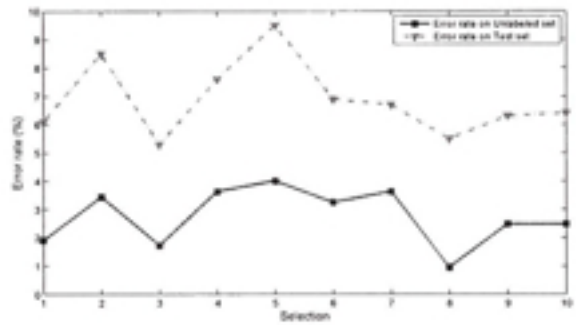


Fig. 9. "Gaus50" dataset: Behavior of Error rate on the unlabeled samples and on the test samples with Algorithm 1.

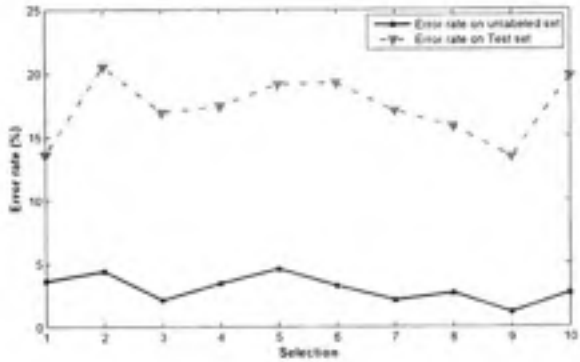


Fig. 10. "Gaus50x" dataset: Behavior of Error rate on the unlabeled samples and on the test samples with Algorithm 1.

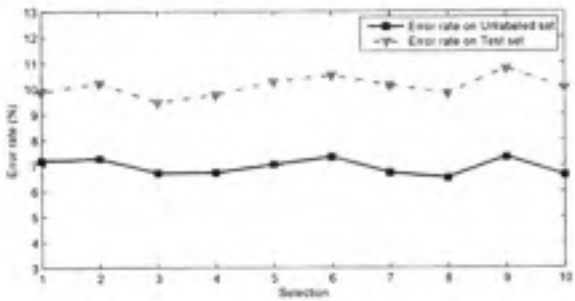


Fig. 11. "USPS" dataset: Behavior of Error rate on the unlabeled samples and on the test samples with Algorithm 1.

It is remarkable that the error rate on the test set varies relative to the error on the labeled set in the case of Algorithm 1 (see Fig. 9, 10 and 11). Also, there are fewer errors on the unlabeled set than on the test set for each problem. However, with Algorithm 2, the errors on the test set and on the unlabeled set are of the same order, meaning that the errors on the unlabeled training set are a reasonable estimate of the true performance of the system on unknown data. This is a remarkable result, considering that, in most semi-supervised algorithms, the similarity between the training algorithm result and the ground truth on the unlabeled set is not the focus, but simply the means to take advantage of the availability of a large dataset. Algorithm 2 gives results that are closer to the ground truth than those of Algorithm 1. So, although Algorithm 1 performs better on unlabeled data than Algorithm 2, the high degree of variance that characterizes this result may suggest a higher sensitivity to overlearning problems. Indeed, the errors of Algorithm

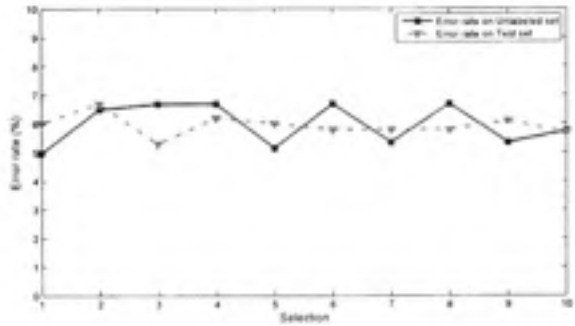


Fig. 12. "Gaus50" dataset: Behavior of Error rate on the unlabeled samples and on the test samples with Algorithm 2.

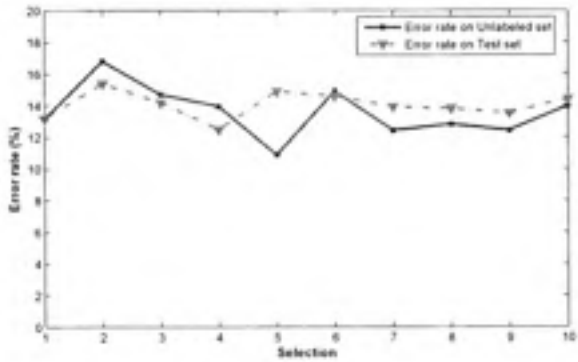


Fig. 13. "Gaus50x" dataset: Behavior of Error rate on the unlabeled samples and on the test samples with Algorithm 2.

2 on the unlabeled set closely reflect the errors of the system on the unseen set. Thus, testing semi-supervised training algorithm on unlabeled data as test set may not be most accurate way to estimate the performance of the resulting classifier (in case of Algorithm 1).

VI. Conclusion

In this paper, we have presented semi-supervised LS-SVM classifier using the transductive inference formulation and proposed two different approaches to train this classifier. We have implemented and evaluated these methods on both artificial and real data. We have obtained encouraging results, which support the usefulness of our algorithms. Compared with the semi-supervised SVM, we have shown that the performance of the semi-supervised LS-SVM was promising. Moreover, we have pointed out here, through various experiments, the impact of

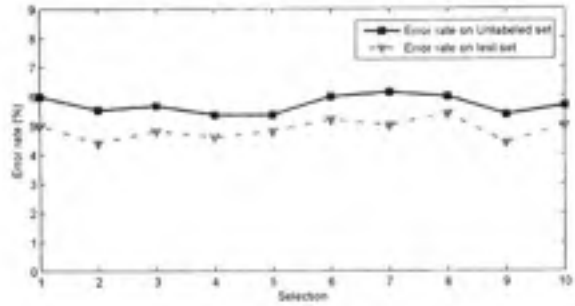


Fig. 14. "Text" dataset: Behavior of Error rate on the unlabeled samples and on the test samples with Algorithm 2.

the hyperparameters on the generalization capacity of our algorithms. The important question from this may become how to select a model when semi-supervised learning is used to build a classifier with a small amount of labeled data. The response to that question will be an interesting direction for our future work.

ACKNOWLEDGMENT

The authors would like to thank the reviewers and the associate editor for their insightful and very useful comments and the NSERC of Canada for their financial support.

REFERENCES

[1] Mathias M. Adankon and Mohamed Cheriet. Learning semi-supervised svm with genetic algorithm. In IEEE, editor, *International Joint Conference in Neural Networks 2007*, pages 1825 – 1830, orlando, FL, 2007.

[2] Mathias M. Adankon and Mohamed Cheriet. Model selection for ls-svm. application to handwriting recognition. In *Eleventh International Conference on Frontiers in Handwriting Recognition*, Montreal, Canada, 2008.

[3] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[4] K. Bennett and A. Demiriz. Semi-supervised support vector machines, 1998.

[5] T. De Bie and N. Cristianini. Convex methods for transduction. In S. Thrun, L. Saul, and B. Scholkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2004.

[6] G. Camps-Valls, T.V. Bandos Marsheva, and D. Zhou. Semi-supervised graph-based hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 45(10):3044–3054, October 2007.

[7] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems 13*, pages 409–415. MIT Press, 2001.

[8] Gavin Cawley. Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In *proceedings IJCNN 2006, Vancouver, Canada*, July 2006.

[9] Gavin C. Cawley and Nicola L. C. Talbot. Improved sparse least-squares support vector machines. *Neurocomputing*, 48:1025–1031, 2002.

[10] Gavin C. Cawley and Nicola L. C. Talbot. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural Networks*, 17:1467–1475, 2004.

[11] O. Chapelle, V. Sindhwani, and S.S. Keerthi. Branch and bound for semi-supervised support vector machines. In *Proceedings of the NIPS 2006*, 2006.

[12] O. Chapelle, V. Sindhwani, and S.S. Keerthi. Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research*, 9:203–233, 2008.

[13] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

[14] W. Chu, C. J. Ong, and S. S. Keerthi. An improved conjugate gradient scheme to the solution of least squares svm. *IEEE Transactions on Neural Networks*, 16(2):498–501, 2005.

[15] Kok Seng Chua. Efficient computations for large least square support vector machine classifiers. *Pattern Recognition Letters*, 24:75–80, 2003.

[16] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Large scale transductive svm. *Journal of Machine Learning Research*, 7:1687–1712, 2006.

[17] B. J. de Kruif and T. J. deVries. Pruning error minimization in least squares support vector machines. *IEEE Transactions on Neural Networks*, 14:696–702, 2003.

[18] G. Fung and O. Mangasarian. Semi-supervised support vector machines for unlabeled data classification. *Optimization Methods and Software*, 15:29–44, 2001.

[19] Glenn Fung and Olvi L. Mangasarian. Proximal support vector machines. In *7 th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 77–86, 2001.

[20] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 529–536. MIT Press, Cambridge, MA, 2004.

[21] M. Hein, J.-Y. Audibert, and U. Von Luxburg. Graph laplacians and their convergence on random neighborhood graphs. *Journal of Machine Learning Research*, 8:1325–1370, 2007.

[22] Licheng Jiao, Liefeng Bo, and Ling Wang. Fast sparse approximation for least square support vector machine. *IEEE Transactions on Neural Networks*, 18(3):685–697, 2007.

[23] Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

[24] Hyunsoo Kim and Haesun Park. Incremental and decremental least squares support vector machine and its application to drug design. *Computational Systems Bioinformatics Conference, International IEEE Computer Society*, 0:656–657, 2004.

[25] J. Luts, J.A.K. Suykens, and S. Van Huffel. Semi-supervised learning: avoiding zero label assumptions in kernel based classifiers. Technical report, Internal Report 07-122, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2007.

[26] B. Scholkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[27] Matthias Seeger. Learning with labeled and unlabeled data. Technical report, Technical Report , Institute for Adaptive and neural Computation, University of Edinburgh, February 2001.

[28] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer-Verlag, New York, 1993.

[29] J.A.K. Suykens, L. Lukas, and L. Vandewalle. Sparse least squares support vector machine classifiers. In *European Symposium of Artificial Neural Networks*, 2000.

[30] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

[31] J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[32] Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[33] T. Van Gestel, J. A. K. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and J. Vandewalle. Benchmarking least squares support vector machine classifiers. *Machine Learning*, 54(1):5–32, 2004.

[34] V. N. Vapnik, O. Chapelle, and J. Weston. Transductive inference for estimating values of functions. *Adavances in Neural Information Processing*, 1999.

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

[36] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. 2003. In 18th Annual Conf. on Neural Information Processing Systems.

[37] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2007. http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf.

ARTICLE 3

SEMI-SUPERVISED LEARNING USING BAYESIAN INFERENCE

M. M. Adankon[1], M. Cheriet[1], and A. Biem[2],
[1] École de Technologie Supérieure, 1100 Notre-Dame Ouest,
Montréal, Québec, Canada H3C 1K3
[2] IBM Research, Watson Research Center,
Yorktown Heights, N.Y. 10598, USA

Cet article est soumis à IEEE Transactions on Pattern Analysis and Machine Intelligence.

Semi-supervised Learning using Bayesian inference

Journal:	*Transactions on Pattern Analysis and Machine Intelligence*
Manuscript ID:	Draft
Manuscript Type:	Regular
Keywords:	classification, semi-supervised learning, bayesian inference

Semi-supervised Learning using Bayesian inference

Mathias M. Adankon, Mohamed Cheriet and Alain Biem

Mathias M. Adankon and Mohamed Cheriet are with the Synchromedia Laboratory for Multimedia Communication in Telepresence at the École de Technologie Supérieure, University of Quebec, 1100 Notre-Dame West, Montreal, Canada, H3C 1K3. (email: mathias@livia.etsmtl.ca, mohamed.cheriet@etsmtl.ca)

Alain Biem is with IBM at the Watson Research Center, New York, USA (email: biem@us.ibm.com)

Abstract

Bayesian reasoning provides an ideal basis for representing and manipulating uncertain knowledge, with the result that many interesting algorithms in machine learning are based on Bayesian inference. In this paper, we use the Bayesian approach with one and two levels of inference to model the semi-supervised learning problem and give its application to the successful kernel classifier, SVM and its variant LS-SVM. Taking advantage of the Gaussian process model, we develop a semi-supervised learning algorithm for Bayesian LS-SVM using our approach based on two levels of inference. Experimental results on both artificial and real pattern recognition problems show the utility of our method.

Index Terms

semi-supervised learning, support vector machine, least-square SVM, Bayesian inference.

I. Introduction

The problem of recognizing patterns through artificial means entails encoding the knowledge inherent in some exemplar data (x, y), where x represents an observation from nature (e.g. pixels of an image) and y is the associated label (e.g. what the pixel represents in the conceptual world). Encoding this knowledge takes the form of a model which can be used to predict the label of previously unseen observations. The most efficient model is the one that implements the Bayes classifier, which assigns the label y to the class of the maximum a posteriori value $P(y|x) = p(x|y)p(y)/p(x)$.

Broadly speaking, a classifier approximating the Bayes classifier can be designed in two ways: the generative approach, and the discriminative approach. The generative approach attempts to estimate $p(x, y)$ for each class of labels y, generally by assuming a parameterized model of the likelihood $p(x|y)$ and using a criterion - such as Maximum Likelihood - to estimate the model's parameters. This is a class-by-class modeling approach. By contrast, the discriminative scheme attempts to directly estimate the boundaries between classes, either by estimating the posterior $p(y|x)$ or by estimating the difference $p(y = 1|x) - p(y = -1|x)$[1], through either log likelihood ratio optimization or other discriminative criteria[8]. Needless to say, an important condition of the success of both schemes remains the availability of a sufficient amount of exemplar data.

[1]assuming a two-class problem with labels 1 and -1

Encoding knowledge using exemplar data (x, y) is referred to as *supervised training*, as that data contains the *ground truth* y, a decision as to what the data represent. The extraction of the ground truth is referred to as *data labeling*, a cumbersome and error-prone process which quickly becomes prohibitive for a large amount of data. For instance, the labeling of handwritten documents, images, and Web pages requires both human expertise and insight, subject to trial and error with damaging effect on the performance of final system. In medicine or biology, testing and very complex experiments generate such a large amount of data that accurate labeling is nearly impossible.

Needless to say, the accuracy of a statistically designed classification system improves with the number of available exemplar data[2]. The contrasting requirements of the availability of a large amount of accurately labeled data and the unavoidable inaccuracies of the labeling process, added to its cumbersome nature, call for the consideration of incomplete or unlabeled data in the design of a pattern recognition system [36]. This is referred to as *semi-supervised learning*[52, 41], the success of which depends on the astute utilization of unlabeled data, according to the target in mind. Generally, the target is either the inference of the Bayes boundaries, in which the ultimate goal is to generate a *comprehensive classifier* that is meant to classify the entire input space, versus *a restrictive classifier* targeted at a specific task (a given set of unlabeled data to be classified).

The utilization of unlabeled data in the learning process calls for key assumptions to be made about the underlying characteristics of the data, even though those assumptions may add further constraints on the choice of model. These assumptions concern the degree to which characteristics of the data, in the form of a data density probability $p(x)$, yield some knowledge of the posterior probability of the label $p(y|x)$.

Assumptions made on data characteristics (listed below) yield different types of classification and optimization schemes [17]:

(i) Cluster assumption: Points in the same cluster are likely to be of the same class. This is a straightforward extrapolation of unsupervised clustering techniques to the semi-supervised arena. It has given rise to various types of classifiers, including margin-maximizing classifiers such as the semi-supervised support vector machine, first introduced as the transductive support

[2]Assuming a correct classifier model

vector machine (TVSM)[31].

(ii) Manifold assumption: The intrinsic dimensionality of the data is in a manifold, on which classification could be carried out more efficiently. The graph-based approach relies on this assumption to link unlabeled instances to their 'close' labeled neighbor in the graph.

The generative approach appears well suited to semi-supervised learning, as the whole dataset (labeled and unlabeled data) can be used to estimate the probability density function $p(x)$, and the labeled portion of the data used to estimate the joint probability $p(x, y) = p(y|x)p(x)$, which provides a natural integration of the data assumptions in the modeling process [13, 32]. However, as previously stated, the generative approach is quite sensitive to model mismatch, and is an indirect path to the primary goal of estimating the boundary between classes.

By contrast, the discriminative method appears to be an attractive choice in terms of performance, but may be less direct in its integration of data assumptions within the modeling and optimization process, depending on the type of algorithm used.

Discriminative training can be implemented in various ways: either within a structure, as with Support Vector Machines, or by choosing a discriminative criterion, as with Maximum Mutual Information or Minimum Classification Error [8]. Structure-based discriminative training opens up various possibilities, including the use of a suitable objective function that enables the integration of data assumptions within the learning process. For instance, margin-maximizing classifiers, such as the Support Vector Machines (SVMs) rely on the cluster assumption and attempt to maximize the classification margin on all data (labeled and unlabeled). Then, there is a range of mixed approaches, such as the self-training, co-training, graph-based methods, etc. In self-training [39, 38], a classifier is first trained with the labeled data and the results are used to classify the unlabeled data. The co-training [10, 36] is based on multi-view learning. The classifiers are trained using each view, and they learn from one another to improve their performance. Graph-based semi-supervised methods [1, 30, 20, 34] define a graph where the nodes represent labeled and unlabeled examples in the dataset. The edges of this graph reproduce the similarity of examples, and they are used to determine the labels.

The aim of this paper is to improve semi-supervised learning through the use of structural discriminative training. We utilize Bayesian inference to infer both the model parameters and the labels for unlabeled data. We describe one- and two-level Bayesian inference schemes and

demonstrate that the one-level Bayesian framework leads, under certain assumptions, to the standard semi-supervised SVM formulation [19]. Furthermore, we develop a complete application for the semi-supervised Least-Squares SVM (LS-SVM) by using two-level inference, which gives promising experimental results. We denote this latest model the *"Bayesian Semi-Supervised Least-Square SVM"* ($BS^2LS-SVM$) where a Gaussian process model is used.

This paper is structured as follows. In section 2, we give an overview of the kernel hyperplane classification framework. In section 3, we describe the Bayesian approach with one- and two-level inference for modeling the semi-supervised learning problem. In sections 4 and 5, we show how to apply our proposed approach to the SVM and its variant, the LS-SVM. In section 6, we present the experimental results on both artificial and real data by using our new semi-supervised training algorithm for the $BS^2LS-SVM$. We conclude the paper in section 7.

II. Kernel-based hyperplane classification

We consider kernel machines as the classifier of choice in this paper: in particular, the SVM and its variants.

Let us assume a binary classification problem and a training set $\mathcal{D}$ comprising ℓ labeled samples $\{(x_1,y_1),\ldots,(x_\ell,y_\ell)\}$ and n unlabeled samples $\{x_1^\star,\ldots,x_n^\star\}$ forming $\mathcal{D}=\{(x_1,y_1),(x_2,y_2),...,(x_\ell,y_\ell),x_1^\star,x_2^\star,...x_n^\star\}$ with $x_i\in\mathbb{R}^d$, $x_i^\star\in\mathbb{R}^d$ and $y_i\in\{-1,1\}$. The purpose of the classification scheme is to enable us to generate a map $f_\theta(x)=y:\mathbb{R}^d\rightarrow\{-1,1\}$ from data to their labels. In machine learning, the complexity of the classification function f_θ greatly influences the performance achieved: a highly complex function may fit training data perfectly, but gives a poor generalization on unseen data [9]. In our case, we consider the classifiers based on a class of hyperplanes:

$$f_\theta(x)=sign[w'x+b] \tag{1}$$

where w' denotes the transpose of w, b is the bias term, and the parameter of the model $\theta=(w,b)$. This family of classifiers divides the input space into two parts: one for positive samples, the labels of which are equal to $y=1$, and the second for negative samples with labels $y=-1$.

In real-world problems, the data are not linearly separable, and so, for many hyperplane classifiers, the kernel trick is used to produce nonlinear boundaries [11]. The idea behind kernels is to map training data nonlinearly into a higher-dimensional feature space via a mapping function Φ and to construct a separating hyperplane in this new space. The construction of the linear decision surface in this feature space only requires the evaluation of dot products $\phi(x_i).\phi(x_j) = k(x_i, x_j)$, where the function $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is called the kernel function [40, 24]. Thus, the classification function becomes:

$$f_\theta(x) = sign\left[w'\phi(x) + b\right] = sign\left[\sum_i \alpha_i y_i k(x_i, x) + b\right] \tag{2}$$

where the $\alpha_i \in \mathbb{R}$ are scalars representing, with b, the classifier parameters in dual space.

A. Review of SVMs

SVMs attempt to resolve the following optimization problem, which expresses the maximization of the margin $2/\|w\|$ and the minimization of the training error:

$$\min_{w,b,\xi} \frac{1}{2}w'w + C\sum_{i=1}^{\ell}\xi_i \tag{3}$$

$$\text{s.t.} \ : y_i[w'\phi(x_i) + b] \geq 1 - \xi_i \quad \forall i = 1, ..., \ell \tag{4}$$

$$\xi_i \geq 0 \quad \forall i = 1, ..., \ell \tag{5}$$

In its primal form, the Lagrangian of this problem is as follows:

$$\mathcal{G} = \frac{1}{2}w'w + C\sum_{i=1}^{\ell}\xi_i - \sum_{i=1}^{\ell}\alpha_i[y_i(w'\phi(x_i) + b) - 1 + \xi_i] - \sum_{i=1}^{\ell}\lambda_i\xi_i \tag{6}$$

with the Lagrange multipliers $\alpha_i \geq 0$ and $\lambda_i \geq 0$ for all $i = 1, ..., \ell$.

Using the conditions of optimality with respect to Lagrange, we obtain the classifier:

$$f(x) = sign[w'\phi(x) + b] = sign[\sum_{i=1}^{\ell}\alpha_i y_i k(x_i, x) + b] \tag{7}$$

with α the solution of (representing the dual problem):

$$maximize : W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{8}$$

$$s.t. : \sum_{i=1}^{\ell} \alpha_i y_i = 0 \quad and \quad 0 \leq \alpha_i \leq C, i = 1, ..., \ell$$

The threshold b is computed by averaging $b = y_j - \sum y_i \alpha_i k(x_i, x_j)$ over all support vectors x_j ($\alpha_j > 0$).

Semi-supervised SVMs are designed to find the hyperplane $< w, b >$ and the labels $y_1^*, \ldots, y_n^*$ of the unlabeled data that maximizes the margin with the minimum error, both on labeled and unlabeled data. Thus, we have:

$$\min_{w,b,\xi,\xi^*,y_1^*,\ldots,y_n^*} \frac{1}{2} w'w + C \sum_{i=1}^{\ell} \xi_i + C^* \sum_{j=1}^{n} \xi_j^* \tag{9}$$

$$\text{s.t.} : y_i[w'\phi(x_i) + b] \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i = 1, ..., \ell \tag{10}$$

$$y_j^*[w'\phi(x_j^*) + b] \geq 1 - \xi_j^*, \quad \xi_j^* \geq 0 \quad \forall j = 1, ..., n \tag{11}$$

The problem expressed in (9) is the semi-supervised SVM formulation used in the literature and many algorithms have been proposed to perform this optimization. This is a difficult problem because of the non-convex nature of the objective function. However, a number of solutions have been proposed with varying degrees of success, including an integer programming method [6], a combinatorial approach [31], and a sequential optimization procedure [26] with good scalable properties, albeit only demonstrated in the linear case. An interesting approach is given in [7], where the non-convex transductive problem is transformed into a convex, semi-definite programming problem. In general, the optimization problem is simplified with an appropriate choice of loss function. Further details on recent semi-supervised SVM optimization techniques can be found in [19].

B. Review of Least Squares SVM

The Least Squares SVM (LS-SVM) is an interesting variant of the SVM proposed by Suykens et al. [45]. This machine is achieved by changing the hinge loss function by a squared-loss

function. It has been shown, through a meticulous empirical study, that the generalization performance of the LS-SVM is comparable to that of the SVM [49]. In addition, the training algorithm of the LS-SVM is greatly simplified, since a linear problem is solved instead of a quadratic programming (QP) problem.

The LS-SVM is derived from the standard Vapnik SVM classifier by transforming the quadratic programming problem into a linear system problem, yielding the optimization

$$\min_{w,b,\xi} \frac{1}{2} w'w + \frac{1}{2} C \sum_{i=1}^{\ell} \xi_i^2 \tag{12}$$

$$\text{subject to } : \xi_i = y_i - [w'\phi(x_i) + b] \quad \forall i = 1, ..., \ell \tag{13}$$

which modifies the original SVM formulation in two respects. First, the inequality constraints with the slack variable ξ_i expressed in (2) are replaced by equality constraints. Second, a squared loss function is considered in the objective function. These two essential modifications simplify the problem, which becomes a linear system.

The Lagrangian of problem (12) is expressed by :

$$\mathcal{S}(w, b, \xi, \alpha) = \frac{1}{2} w'w + \frac{1}{2} C \sum_{i=1}^{\ell} \xi_i^2 - \sum_{i=1}^{\ell} \alpha_i \{y_i - [w'\phi(x) + b] - \xi_i\}$$

where α_i are Lagrange multipliers, which can be positive or negative because of equality constraints.

The conditions for optimality yield

$$\begin{cases} \frac{\partial \mathcal{S}}{\partial w} = 0 & \Rightarrow \quad w = \sum_{i=1}^{\ell} \alpha_i \phi(x_i) \\ \frac{\partial \mathcal{S}}{\partial b} = 0 & \Rightarrow \quad \sum_{i-1}^{\ell} \alpha_i = 0 \\ \frac{\partial \mathcal{S}}{\partial \xi_i} = 0 & \Rightarrow \quad \alpha_i = C\xi_i, \quad \forall i = 1, ..., \ell \\ \frac{\partial \mathcal{S}}{\partial \alpha_i} = 0 & \Rightarrow \quad \xi_i = y_i - [w'\phi(x_i) + b] \quad \forall i = 1, ..., \ell \end{cases}$$

We note that the system arising from these conditions (Karush-Kuhn-Tucker conditions) is linear, and that its solution is found by solving the system of linear equations expressed in the

following matrix form:

$$\begin{pmatrix} K + C^{-1}I & \vec{1}' \\ \vec{1} & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ b \end{pmatrix} = \begin{pmatrix} Y \\ 0 \end{pmatrix} \tag{14}$$

where:

$K_{ij} = k(x_i, x_j)$

$Y = (y_1, ..., y_\ell)'$

$\alpha = (\alpha_1, ..., \alpha_\ell)'$

$\vec{1} = (1, ..., 1)$

Several methods have been developed for supervised LS-SVMs, such as a training algorithm [45, 22, 21], model selection [16, 14, 2], and sparseness [44, 15, 25, 29]. However, to our knowledge, only the work of Juan et al. [34] on the semi-supervised LS-SVM is available. In this work, the authors introduced an extra variable and additional equality constraints into the classifier formulation in order to avoid the zero label assumption.

III. Bayesian inference for semi-supervised learning

Bayesian approaches in statistics are well suited to problems where model design and selection are difficult due to the presence of abundant latent variables [35]. As such, Bayesian inference provides room to integrate unlabeled data as a latent variable in the modeling process.

Bayesian inference starts with the formulation of a hypothesis or model, which is refined incrementally as more knowledge of the situation becomes available. A prior distribution over the unknown parameters of the model expresses our knowledge or assumptions about the situation before the data are seen. After observing some of the data, the Bayes' theorem is used to compute a posterior distribution of the unknowns.

In this work, we apply Bayesian inference to the estimation of both the parameter θ of the model and the unknown labels $z = (y_1^*, y_2^*, ..., y_n^*)$.

A. Maximum a posteriori: One-level inference

Given a hypothesis or model structure $\mathcal{H}$, the goal is to infer the model parameters θ and the labels $z = (y_1^*, y_2^*, ..., y_n^*)$ by maximizing the posterior distribution, which is expressed as follows:

$$P(\theta, z|\mathcal{D}) = \frac{P(\mathcal{D}, \theta, z)}{P(\mathcal{D})} \tag{15}$$

where $P(\mathcal{D}, \theta, z)$ is the joint probability of the data $\mathcal{D}$, the model's parameters θ and the unknown label [3].

Using the knowledge about the model's parameters θ with the prior $P(\theta)$ and the likelihood $P(\mathcal{D}, z|\theta)$, the posterior distribution becomes:

$$P(\theta, z|\mathcal{D}) = \frac{P(\mathcal{D}, z|\theta)}{P(\mathcal{D})} P(\theta) \tag{16}$$

Maximizing Eq. 16 determines both the parameter θ of the model and the labels z of the unlabeled data. As the prior probability $P(\mathcal{D})$ is a simply a normalizing constant, this is equivalent to maximizing the logarithm of the numerator:

$$\mathcal{L}(\theta, z) = log[P(\mathcal{D}, z|\theta)P(\theta))] \tag{17}$$

The most important question is how to optimize this expression, where the search spaces are not the same, i.e. continuous [4] for the parameter θ of the model and discontinuous [5] for the labels z of the unlabeled data. Equally important is the choice of the form of the priors $p(\theta)$.

B. Two-level inference

A Multi-level inference process is applied in order to model one of the variables at each step by decoupling the parameter θ of the model and the labels z of the unlabeled data.
Application of the Bayes rule at the first level of inference gives

[3] As all probabilities are conditioned on the model structure $\mathcal{H}$, we omit explicit reference to $\mathcal{H}$.

[4] This is dependent on the prior distribution of θ.

[5] For example, in binary classification the elements of z are -1/1 or 0/1.

$$P(\theta|\mathcal{D}, z) = \frac{P(\mathcal{D}, z|\theta)}{P(\mathcal{D}, z)} P(\theta) \tag{18}$$

where $P(\mathcal{D}, z|\theta)$ is the conditional probability of the data $\mathcal{D}$ and the label z given the model parameter θ , and $P(\theta)$ is the prior probability of the model. The evidence at this level of inference is a joint probability of the $\mathcal{D}$ and z and it is obtained by integrating the numerator over the parameter space:

$$P(\mathcal{D}, z) = \int_{\Theta} P(\mathcal{D}, z|\theta) P(\theta) d\theta \tag{19}$$

This calls for a second-level inference regarding the labels z as

$$P(z|\mathcal{D}) = \frac{P(\mathcal{D}, z)}{P(\mathcal{D})} \tag{20}$$

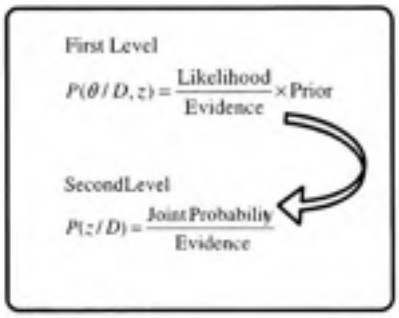


Fig. 1. Illustration of two-level inference for semi-supervised learning.

Fig. 1 illustrates the linkage between the two levels of inference. The joint probability used to express the posterior of equation (20) represents the evidence in Bayes' rule at the first level of inference and is computed by using equation (19). Then, we obtain

$$P(z|\mathcal{D}) \propto P(\mathcal{D}, z) \tag{21}$$

$$\propto \int_{\Theta} P(\mathcal{D}, z|\theta) P(\theta) d\theta \tag{22}$$

In short, the semi-supervised learning process based on two-level inference consists of:
(i) finding the vector z that maximizes Eq. 22; and
(ii) using this value in Eq. 18 to estimate the parameter θ.

IV. Application of one-level inference to SVMs

This section demonstrates that the application of one-level inference to SVM leads to semi-supervised SVM formulation [19], also referred to as the transductive SVM, or TSVM [50]. Using the probabilistic interpretation of SVMs proposed in [43], where the prior distribution is Gaussian (margin-maximization) and the hinge loss function represents the likelihood, we have

$$P(\mathcal{D}, z|\theta) \propto \prod_i Q(y_i|x_i,\theta)Q(x_i) = \prod_i exp\big(-Cl(y_i[w'\phi(x_i)+b])\big)Q(x_i) \tag{23}$$

and

$$P(\theta) \propto exp\big(-\frac{1}{2}\|w\|^2\big) \tag{24}$$

with $\theta = (w, b)$ and $l(t) = \max(0, 1-t)$ is the hinge loss function.

We need to mention here as pointed out in [43] that the likelihood function is **not normalized** because $Q(y_i = +1|x_i,\theta) + Q(y_i = -1|x_i,\theta)$ is not equal to one for each sample.

Considering the previous Bayesian interpretation for SVM, we obtain according Eq.(17)

$$\mathcal{L}(\theta, z) = \mathcal{L}(w, b, z) \propto -\frac{1}{2}\|w\|^2 - C\sum l(y_i[w'\phi(x_i)+b]) \tag{25}$$

Minimizing $-\mathcal{L}(w, b, z)$ gives

$$\min_{w,b,y_1^*,y_2^*,\ldots,y_n^*} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell} l(y_i[w'\phi(x_i)+b]) + C^*\sum_{j=1}^{n} l(y_j^*[w'\phi(x_j^*)+b]) \tag{26}$$

The optimization problem expressed by (26) yields the TSVM proposed by Vapnik in [50] and described earlier. In addition, this formulation, which maximizes the margin on both labeled and unlabeled data, provides a decision boundary which lies in a low-density region implementing the cluster assumption for semi-supervised learning [20].

Minimizing (26) over θ and z is very hard, so one could include prior knowledge about the label z, as is done in many semi-supervised SVM algorithms, by using a ratio constraint between positive and negative data, also called a balancing constraint [31, 20, 18]. This constraint enforces a fraction, R, of unlabeled data to be assigned to the positive class:

$$R = \frac{1}{n}\sum_{j=1}^{n} max(y_j^*, 0) = \frac{1}{2}\left[1 + \frac{1}{n}\sum_{j=1}^{n} y_j^*\right] \tag{27}$$

Similarly, we can apply one-level inference (MAP estimation) to the LS-SVM classifier and deduce the semi-supervised LS-SVM formulation by replacing the SVM hinge loss function by the quadratic one. Thus, this work can be extended to other classifiers to design new semi-supervised training algorithms.

V. Application of Two-Level Inference

It is difficult to apply two-level inference with standard SVM classifiers, since the integration expressed by (19) over θ-space cannot be performed analytically. By contrast, the Bayesian learning formulation of the LS-SVM leads to an analytical solution of the integral (19), as shown below.

While the LS-SVM has proved to be a successful alternative to standard SVMs, the number of applications in semi-supervised settings has been limited. In this section, we propose to develop the semi-supervised LS-SVM using our Bayesian framework with two-level inference. As stated above, we refer to this algorithm as the $BS^2LS - SVM$.

Similar to the Suykens et al. formulation [46, 45, 48, 51], the likelihood and the prior of the LS-SVM Bayesian formulation are as follows:

$$P(\mathcal{D}, z|\theta) = \prod_i \sqrt{\beta/2\pi} exp\big(-\frac{1}{2}\beta e_i^2\big) Q(x_i) \tag{28}$$

with

$$e_i = y_i - [w'\phi(x_i) + b] \tag{29}$$

$$P(\theta) = P(w,b) = \Big(\frac{\mu}{2\pi}\Big)^{n_h/2} exp\big(-\frac{1}{2}\mu w'w\big)\frac{1}{\sqrt{2\pi}\sigma_b} exp\big(-\frac{1}{2\sigma_b^2}b^2\big) \tag{30}$$

where n_h denotes the dimension of the feature space for the mapping function Φ.

The prior corresponds to the maximization (regularization) term and the sum of the squared error variables corresponds to the likelihood. Note that the likelihood function is not normalized like in SVM case if we consider classification view where $y \in \{+1, -1\}$. When the prior distribution of bias b approximates a uniform distribution when $\sigma_b \longrightarrow \infty$ (use of improper prior), the total prior of LS-SVM model becomes

$$P(w,b) = \Big(\frac{\mu}{2\pi}\Big)^{n_h/2} exp\big(-\frac{1}{2}\mu w'w\big) \tag{31}$$

Defining

$$\tilde{\phi}(x_i) = (\phi(x_i), 1) \tag{32}$$

and

$$J = \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} \tag{33}$$

where I is the identity matrix,

we can rewrite the prior and the likelihood as

$$P(\mathcal{D}, z|\theta) = \prod_i \sqrt{\beta/2\pi} exp\big(-\frac{1}{2}\beta(y_i - \theta'\tilde{\phi}(x_i))^2\big) \times \prod_i Q(x_i) \tag{34}$$

and

$$P(\theta) = \Big(\frac{\mu}{2\pi}\Big)^{n_h/2} exp\big(-\frac{1}{2}\mu\theta' J\theta\big) \tag{35}$$

Before progressing in our development, it is important to mention that Bayesian LS-SVM classifier is a Gaussian process model. Thus, we deal with the Gaussian distributions and take advantage of the model. For more details on Gaussian process for machine learning, see [37].

With the previous expressions of the prior and the likelihood, we obtain

$$\begin{aligned} P(\mathcal{D}, z|\theta)P(\theta) &= \prod_i \sqrt{\beta/2\pi} exp\Big(-\frac{1}{2}\beta(y_i - \theta'\tilde{\phi}(x_i))^2\Big)\Big(\frac{\mu}{2\pi}\Big)^{n_h/2} exp\Big(-\frac{1}{2}\mu\theta' J\theta\Big) \times \prod_i Q(x_i) \\ &= \Big(\frac{\beta}{2\pi}\Big)^{(\ell+n)/2}\Big(\frac{\mu}{2\pi}\Big)^{n_h/2} exp\Big(-\frac{1}{2}\mu\theta' J\theta - \frac{1}{2}\beta\sum_i (y_i - \theta'\tilde{\phi}(x_i))^2\Big) \times \prod_i Q(x_i) \\ &= \Big(\frac{\beta}{2\pi}\Big)^{(\ell+n)/2}\Big(\frac{\mu}{2\pi}\Big)^{n_h/2} exp\Big(-\frac{1}{2}\mu\theta' J\theta \\ &- \frac{1}{2}\beta\sum_i [y_i^2 - 2y_i\theta'\tilde{\phi}(x_i) + \theta'(\tilde{\phi}(x_i)\tilde{\phi}(x_i)')\theta]\Big) \times \prod_i Q(x_i) \end{aligned}$$

After some manipulations, the following result is obtained:

$$P(\mathcal{D}, z|\theta)P(\theta) = \Lambda exp\Big[-\frac{1}{2}(\theta - \theta_{MAP})' H(\theta - \theta_{MAP})\Big] exp\Big[\frac{1}{2}F_y' H^{-1} F_y\Big] \times \prod_i Q(x_i) \quad (36)$$

where

$$\Lambda = \Big(\frac{\beta}{2\pi}\Big)^{(\ell+n)/2}\Big(\frac{\mu}{2\pi}\Big)^{n_h/2} exp\Big(-\frac{1}{2}\beta(\ell+n)\Big) \quad (37)$$

$$F_y = \beta\sum_i y_i\tilde{\phi}(x_i) \quad (38)$$

$$H = \beta\sum_i (\tilde{\phi}(x_i)\tilde{\phi}(x_i)') + \mu J \quad (39)$$

$$\theta_{MAP} = H^{-1}F_y \quad (40)$$

Therefore, the value of integral (19) is

$$\int P(\mathcal{D}, z|\theta)P(\theta)d\theta = \Lambda\sqrt{(2\pi)^{n_h+1} det H^{-1}} exp\Big[\frac{1}{2}F_y' H^{-1} F_y\Big] \times \prod_i Q(x_i) \quad (41)$$

Considering Eq. 22, we have

$$P(z|\mathcal{D}) \propto exp\Big[\frac{1}{2}F_y' H^{-1} F_y\Big] \tag{42}$$

Maximizing this posterior, $P(z|\mathcal{D})$, with respect to the labels of the unlabeled data is equivalent to finding the labels $y_1^*, y_2^*, ..., y_n^*$ that maximize $F_y' H^{-1} F_y$.

$$\max_{y_1^*, y_2^*, ..., y_n^*} \Big(\sum_{i=1}^{\ell} y_i \tilde{\phi}(x_i) + \sum_{j=1}^{n} y_j^* \tilde{\phi}(x_j^*) \Big)' H^{-1} \Big(\sum_{i=1}^{\ell} y_i \tilde{\phi}(x_i) + \sum_{j=1}^{n} y_j^* \tilde{\phi}(x_j^*) \Big) \tag{43}$$

with $y_j^* \in \{-1, 1\}$.

Based on the theoretical study described in the previous paragraphs, the training algorithm for the semi-supervised LS-SVM is derived as follows:

- Find the labels of the unlabeled samples that maximize $F_y' H^{-1} F_y$;
- Compute the model parameters for the classifier.

The details of the complete algorithm are shown in algorithm 1 where certain heuristic is used to reduce the number of the unlabeled samples before solving (43). Problem (43) can be solved with any Integer Quadratic Programming Solver, such as CPLEX [28]. In practice, we can obtain an unbalanced solution, for example $y_i = 1, \forall i$. Thus, we solve (43) under the balancing constraint used in many semi-supervised SVM algorithms[6]. Explicitly, we propose the following integer QP problem under linear constraint.

$$\max_{y_1^*, y_2^*, ..., y_n^*} \Big(\sum_{i=1}^{\ell} y_i \tilde{\phi}(x_i) + \sum_{j=1}^{n} y_j^* \tilde{\phi}(x_j^*) \Big)' H^{-1} \Big(\sum_{i=1}^{\ell} y_i \tilde{\phi}(x_i) + \sum_{j=1}^{n} y_j^* \tilde{\phi}(x_j^*) \Big) \tag{44}$$

$$\text{s.t. } R_1 \leq \frac{1}{2}\Big[1 + \frac{1}{n}\sum_{j=1}^{n} y_j^*\Big] \leq R_2 \tag{45}$$

$$y_j^* \in \{-1, 1\} \tag{46}$$

We can notice that, in our context, we relax the balancing constraint by using two inequalities instead of equality used in almost S^3VM training algorithms.

[6]We expand our explanation of this constraint in Eq. 27.

Algorithm 1 Semi-supervised LS-SVM with two-level inference: $BS^2LS-SVM$ algorithm

Input Labeled samples $\{X, Y\}$ and Unlabeled samples X^*

Output Hyperplane $< w, b >$

- Select the kernel function and kernel parameters
- Set the hyperparameters
- Train the machine with labeled samples $\{X, Y\}$
- Select the unlabeled samples which are close to labeled samples

$S = \{x^* \in X^* | \min_{x \in X} \|x^* - x\| \leq d_0\}$

- Classify and label the samples of S whose output are most confident, $f(x) \geq \epsilon$
- Solve the problem expressed in (44)
- Compute the model parameter θ with equation (40)
- Return the classifier parameters $< w, b >$

In this work, we apply our formulation to LS-SVMs which are a MAP approximation to Gaussian process models. Thus, it is important to stress existing work on semi-supervised technique using Gaussian process model.

Lawrence and Jordan [33] have proposed a probabilistic approach to learning a Gaussian process classifier with both labeled and unlabeled data. They have derived their method from an ordered categorical model containing three categories and termed it *null category noise models (NCNM)*. Their approach implements the cluster assumption within a probabilistic framework and imposes that no unlabeled data should come from the region around the decision boundary.

Sindhwani and al. [42] applied graph-based construction of semi-supervised kernel for Gaussian processes, in order to draw the benefits of a Bayesian approach. Like many semi-supervised learning algorithms, the data geometry is modeled as a graph whose vertices are both the labeled and unlabeled data, and was applied to semi-supervised Gaussian process classifier.

Recently, Patel and al.[4] have proposed new algorithms for semi-supervised classification where they combine maximum margin clustering principle with sparse Gaussian process re-

gression models and derive three algorithms, SSuSVR (semi-supervised using support vector regression), SSuGPR (semi-supervised using Gaussian process regression) and SSuGPS (semi-supervised using sparse Gaussian process regression). Also, we may cite other Bayesian methods which incorporate unlabeled data [53, 5]. However, these approaches have been designed for transductive Bayesian learning, since, they give prediction for only unlabeled data and not for unseen data.

Although our semi-supervised approach for LS-SVM, $BS^2LS-SVM$ uses Gaussian process model in order to draw the benefits of a Bayesian methodology, our method differs from the previous methods in many points. A major difference is that our approach uses two levels of inference, one for the label of unlabeled data and the second for the model parameter. In general, we note that the main difference between these methods comes from the way that the knowledge about the labels of unlabeled samples are modeled. Also, our learning algorithm is not derived from the expectation-maximization (EM) algorithm principle, where the labels of unlabeled data and the classifier parameters are estimated at each step iteratively until convergence.

VI. Experiments

A. Datasets

1) Gaus50: We consider a two-class problem in a 50-dimensional input space like the artificial datasets used in [27]. The data of each class are generated with equal probability from a Gaussian distribution with a unit covariance matrix. The mean of the Gaussian is $(a, a, ..., a)$ for class 1 and $-(a, a, ..., a)$ for class 2. In order to obtain a Bayes error equal to 5%, the parameter a is fixed to 0.23. We generated 550 samples for training and 1000 samples for testing.

2) Gaus50x: We have here another artificial two-class problem in a 50-D space but with multimodal distribution. The data in each class are generated with equal probability from a Gaussian mixture distribution with significant overlap. The two classes have equal priors and their class-conditional distributions are, respectively, $p(x|y=1) = 0.49\mathcal{N}(\mu_1, I) + 0.51\mathcal{N}(\mu_2, I)$ and $p(x|y=-1) = 0.49\mathcal{N}(-\mu_1, I) + 0.51\mathcal{N}(-\mu_2, I)$ where $\mu_1 = (0.25, 0.25, ..., 0.25, 0.25)$, $\mu_2 = (0.25, 0.25, ..., -0.25, -0.25)$ with 25 negatives entries, $N(\mu, I)$ is a Gaussian density

function with mean vector μ, and I is the identity matrix for its covariance matrix. We generated 550 samples for training and 1000 samples for testing.

3) Text: Text document classification is a problem involving high dimensionality. We used the *Mac* and *Windows* classes taken from the 20 news group dataset, with 7511 dimensions, as preprocessed in [47]. We divided the dataset into two subsets: 1446 samples for training and 500 for testing.

4) USPS: USPS is the well-known US Postal Service handwritten digits recognition corpus. The digits are represented by normalized grey scale images of size 16×16. The learning dataset contains 7291 samples for training, while the testing dataset consists of 2007 other samples. We have a multi-class problem with 10 classes. We then trained 10 machines using the *one-against-all* strategy.

B. Comparison between the supervised and semi-supervised LS-SVMs

To quantify the effect of semi-supervised learning, we ran a set of experiments using the LS-SVM classifier trained with either labeled data or a mix of labeled and unlabeled data. We randomly selected 5% of the training set to form the labeled data, the remaining data consisting of the unlabeled samples. We repeated this operation 10 times, each time testing our semi-supervised algorithm based on two-level Bayes inference and the single LS-SVM trained only with labeled data. We used the RBF or Gaussian kernel with parameter $\gamma = 1/\sigma^2 = 0.02$ and $\mu = \beta = 1$ but with the dataset *Gaus50x* $\beta = 0.1$. The resolution of (44) is done in the feature space[7]. Then, we used the KPCA (Kernel Principal Component Analysis) [40] strategy in order to develop a nonlinear classifier in the primal space by selecting the first principal components. The hyperparameters include the number of the principal components, are selected by minimizing classification error on validation set. The results are shown in Tables I, II, and III, where we add the test error rate obtained by the supervised LS-SVM classifier with labels which are all true.

We have two main comments on the results reported in Tables I and II. First, semi-supervised learning gives good results. Adding unlabeled data improves the generalization capacity of the

[7]We need the explicit expression of the function Φ. So, Using KPCA, we have a good approximation in feature space by selecting the first principal components

TABLE I

ERROR RATE (%) ON THE "GAUS50" DATASET

Selection	LS-SVM with only labeled data	Our Semi-supervised LS-SVM ($BS^2LS-SVM$)	Supervised LS-SVM with all-true labels
#1	10.60	5.90	
#2	26.40	7.00	
#3	20.10	5.60	
#4	34.80	6.20	
#5	36.50	6.60	5.90%
#6	45.90	6.10	
#7	12.40	5.80	
#8	10.30	5.90	
#9	11.40	6.30	
#10	11.80	5.80	
Mean	22.02	6.12	5.90%

classifier, and sometimes we approach the Bayes error. For example, the Bayes error for the artificial problem is 5%, and the semi-supervised algorithm gives a 5.6% error rate for selection #3. Second, we note that the semi-supervised algorithm is strong, even though the distribution information given by the labeled samples is far from reality for selections #4, #5, and #6 in Table I , for selections #2, #4, #5, and #6 in Table II and all the selections in Table III. In these cases, the classifier resulting in the supervised learning gives a high error rate, while the classifier obtained from semi-supervised learning gives better result.

In addition, compared with supervised learning, where we used all available data with their true labels, our semi-supervised algorithm gives promising results: for the Gaus50 dataset, 6.12% versus 5.90%; for the Gaus50x dataset, 14.87% versus 12.20%; and for the Text dataset, 5.22% versus 3.2%. These results confirm the effectiveness of our approach in making use of unlabeled samples for an efficient approximation of the Bayes classifier performance.

TABLE II

ERROR RATE (%) ON THE "GAUS50X" DATASET

Selection	LS-SVM with only labeled data	Our Semi-supervised LS-SVM ($BS^2LS-SVM$)	Supervised LS-SVM with all-true labels
#1	17.30	13.80	
#2	43.50	17.00	
#3	28.70	14.50	
#4	44.10	13.80	
#5	47.10	14.40	12.20%
#6	48.60	18.40	
#7	19.70	14.20	
#8	24.30	13.70	
#9	16.00	13.60	
#10	26.70	15.30	
Mean	31.60	14.87	12.20%

C. Comparison of our method with the semi-supervised SVM

In this section, we compare our semi-supervised algorithm with the semi-supervised SVM (S^3VM). The following S^3VM algorithms were tested:

- $S^3VM-light$ [31], which uses a combinatorial search over the unlabeled data, while keeping the ratio of positive versus negative the same as the ratio in the labeled set;
- S^3VM-GA, which uses the Genetic Algorithm to find the optimal solution over the labeled and unlabeled dataset [3];
- $S^3VM-CCCP$ (concave-convex procedure), which decomposes a non-convex loss function into a convex part and a concave part, and where the convex part is iteratively approximated by a Taylor expansion [23].

We used all four datasets described in section 4.1 and the same experimental setup as described in [3]: for each dataset, we labeled **10%** of the training samples and kept the rest as unlabeled

TABLE III

ERROR RATE (%) ON THE "TEXT" DATASET

Selection	LS-SVM with only labeled data	Our Semi-supervised LS-SVM ($BS^2LS-SVM$)	Supervised LS-SVM with all-true labels
# 1	47.80	5.40	
# 2	50.40	5.00	
# 3	50.40	5.40	
# 4	50.20	5.20	
# 5	50.40	5.60	3.2%
# 6	49.60	4.80	
# 7	49.60	5.60	
# 8	50.40	4.80	
# 9	50.20	4.80	
# 10	49.60	5.60	
Mean	49.94	5.22	3.2%

data; the error rate is computed on the test set, which is unseen data during the training process. The hyperparameter values are fixed by a cross-validation strategy, and are reported in Table IV[8].

The results of comparing our semi-supervised algorithm, $BS^2LS-SVM$, with other semi-supervised algorithms[9] are reported in Table V. We note that the semi-supervised LS-SVM, $BS^2LS-SVM$, achieved a good performance and its generalization performance is comparable to that of the semi-supervised SVM.

[8]We used also cross-validation for selecting the hyperparameters for the other semi-supervised algorithms

[9]We tested the semi-supervised SVM algorithms, the code of which is available on website. We did not find a semi-supervised LS-SVM training algorithm.

TABLE IV

HYPERPARAMETER VALUES USED FOR TRAINING OUR SEMI-SUPERVISED ALGORITHM. n_{KPCA} CORRESPONDS TO THE NUMBER OF PRINCIPAL COMPONENTS EXTRACTED WITH KPCA.

	γ	μ	β	n_{KPCA}
Gaus50	0.02	1	1	100
Gaus50x	0.02	1	0.1	100
Text	0.02	1	1	1446
USPS	0.078	1	10	300

TABLE V

COMPARISON BETWEEN THE SEMI-SUPERVISED SVMS AND THE SEMI-SUPERVISED LS-SVM, $S^3VM-light$ [31], $S^3VM-CCCP$ [23] AND S^3VM-GA [3] LEARN THE SEMI-SUPERVISED SVM, WHILE THE $BS^2LS-SVM$ LEARNS THE SEMI-SUPERVISED LS-SVM USING BAYES INFERENCE. IN THE TABLE, WE REPORT THE ERROR RATE ON THE TEST SET.

	$S^3VM-light$	$S^3VM-CCCP$	S^3VM-GA	$BS^2LS-SVM$
Gaus50	8.80%	7.70%	5.70%	**5.20%**
Gaus50x	16.30%	15.50%	14.70%	**12.00%**
Text	5.40%	5.45%	**4.00%**	4.80%
USPS	14.70%	7.32%	7.62%	**6.92 %**

D. Comparison of our method with other semi-supervised algorithms using Gaussian process

To compare the generalization performances of the $BS^2LS-SVM$ with other semi-supervised algorithms based on Gaussian process, we conducted a experiment on six datasets from the UCI benchmark repository [10] and followed the experimental setup used in [4]. Each dataset was randomly divided into three parts: labeled, unlabeled and test sets. This procedure was repeated ten times for each dataset which is described in Table VI. We tested our algorithm of these

[10]We use these datasets in order to compare our results with those found in the literature.

problems and the results are summarized in Table VII where we reported the results obtained with the various semi-supervised algorithms using Gaussian process in [4].

TABLE VI

DESCRIPTION OF UCI DATASETS. N_l, N_u AND N_t DENOTE THE NUMBER OF LABELED, UNLABELED AND TESTING SAMPLES RESPECTIVELY. THESE NUMBERS ARE SET ACCORDING EXPERIMENTAL SETUP IN [4].

Dataset	N_l	N_u	N_t
Ionosphere	36	140	175
Pima	58	324	384
Splice	50	450	500
Thyroid	12	96	107
Waveform	50	1950	3000
Ringnorm	100	400	500

From Table VII, it is clear that our method performs better than the other techniques on many datasets. Specially, on the datasets like Thyroid and Ringnorm, our algorithm outperforms significantly the others algorithms reported in [4] which used Gaussian process model. However, we note that the performance of our method is modest when the overlap between the classes is extremely important (e.g. Pima dataset).

E. Impact of hyperparameters

To train the semi-supervised LS-SVM using the Bayes formulation with two-level inference, the hyperparameters (kernel, β, and μ) need to be set to their optimal values. We know that the choice of kernel corresponds to the selection of a learning function space. In fact, the kernel determines the functional form of all the possible solutions. Thus, the choice of kernel is very important in building a good classifier [40].

In this section, we focus on the hyperparameters β and μ, which control the trade-off between maximizing the margin and minimizing the total cost error on the training data.

TABLE VII

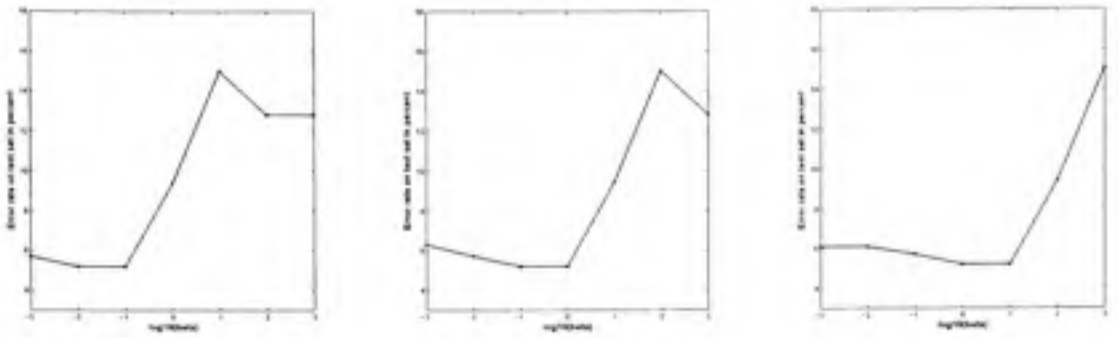
COMPARISON OF OUR METHODS WITH OTHER TECHNIQUES USING GAUSSIAN PROCESS AND TESTED IN [4]. SSUGPR AND SSUGPS DENOTE SEMI-SUPERVISED USING GAUSSIAN PROCESS REGRESSION AND USING SPARSE GAUSSIAN PROCESS REGRESSION RESPECTIVELY. THE RESULTS OF SSUGPR, SSUGPS AND NCNM COME FROM [4]. TEST ERROR IN PERCENT (%) OVER 10 REALIZATIONS.

Datasets	SSuGPR	SSuGPS	NCNM	Our method
Ionosphere	15.20 ± 5.66	11.03 ± 4.92	19.54 ± 6.11	**10.45 ± 4.45**
Pima	**28.98 ± 4.01**	33.44 ± 5.80	29.45 ± 4.43	29.78 ± 1.95
Splice	18.68 ± 2.87	17.06 ± 2.42	17.52 ± 3.08	**16.80 ± 2.54**
Thyroid	16.17 ± 7.72	15.42 ± 7.76	19.35 ± 4.36	**7.09 ± 1.87**
Waveform	12.82 ± 1.75	13.19 ± 1.91	15.30 ± 2.70	**12.28 ± 1.60**
Ringnorm	5.40 ± 1.74	6.26 ± 1.70	17.72 ± 5.40	**2.24 ± 0.55**

We perform various experiments to show the influence of these hyperparameters on the generalization capacity of the classifier built in semi-supervised context.

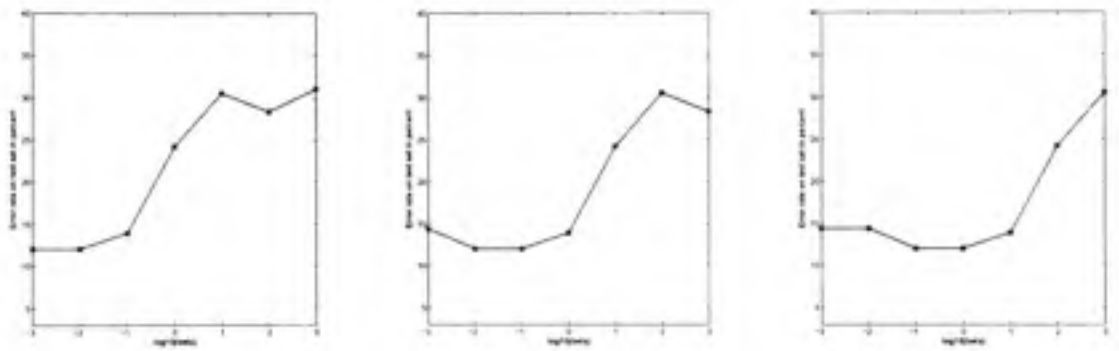
We set μ equal to 0.1, 1, and 10 respectively. Then, for each fixed value of μ, we vary β, train the classifier, and compute the test error in order to quantify the impact of the hyperparameter. The results obtained, shown in Figures 2, 3, 4, and 5, reveal that it is important to choose an optimal value for both μ and β in order to achieve good performance. Also, we note that the ratio μ/β is very useful in fixing the optimal value for the hyperparameters [48]. In our example, it is remarkable that we obtained the best classifier performance when the ratio is equal to 1 or 10 for the Gaus50 dataset (see Fig. 2), 10 or 100 for the Gaus50x dataset (see Fig. 3), 1 for the Text dataset (see Fig. 4), and 0.1 for the USPS dataset (see Fig. 5).

In order to choose the best hyperparameters for the model, we can used the third level of inference, as was done in [12, 48]. However, in our case, we have a computation limitation problem, and also we must make approximations because the integral giving the evidence at the second level of inference cannot be computed analytically.



$\mu = 0.1$ $\mu = 1$ $\mu = 10$

Fig. 2. Impact of hyperparameters in the semi-supervised LS-SVM with the Gaus50 dataset: we set μ equal to 0.1, 1, and 10 respectively, and vary β.

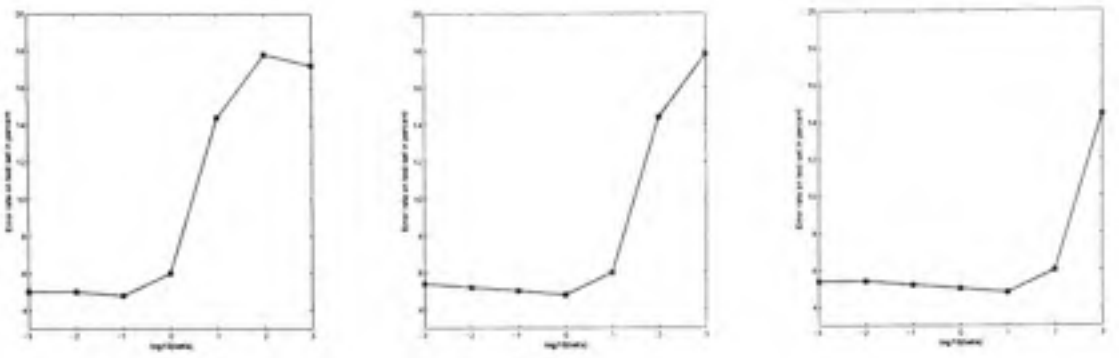


$\mu = 0.1$ $\mu = 1$ $\mu = 10$

Fig. 3. Impact of hyperparameters in the semi-supervised LS-SVM with the Gaus50x dataset: we set μ equal to 0.1, 1, and 10 respectively, and vary β.

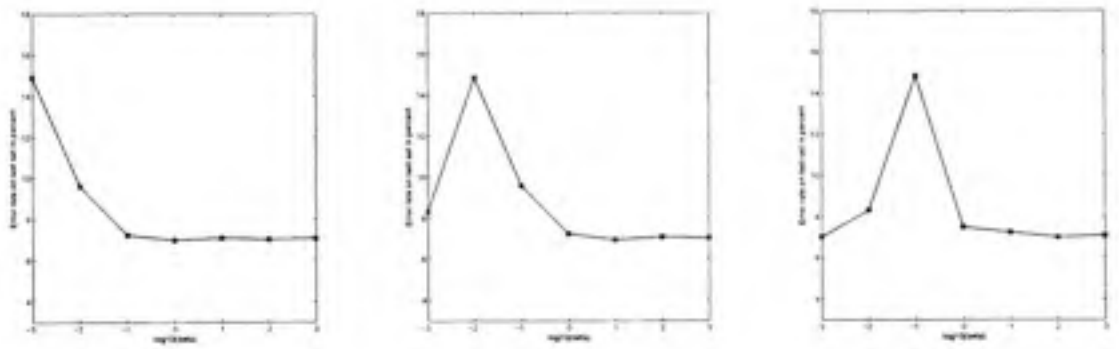
F. Impact of the ratio of the labeled samples

We also tested the behavior of our algorithm with respect to the ratio N_l/N_u, where N_u represents the number of unlabeled samples and N_l the number of labeled samples. Figure 6 shows the variation of the test error when we increased the number of labeled samples in the training data. First, we see that the test error drops quickly as the ratio N_l/N_u increases between 1% and 10%. Since we have more information about the sample labels, the generalization error is improved. Second, we note that the test error rate is less dependent on the ratio when this ratio is greater than 10%. This is a good property of our algorithm, because, for certain semi-supervised



$\mu = 0.1$ $\mu = 1$ $\mu = 10$

Fig. 4. Impact of hyperparameters in the semi-supervised LS-SVM with the Text dataset: we set μ equal to 0.1, 1, and 10 respectively, and vary β.



$\mu = 0.1$ $\mu = 1$ $\mu = 10$

Fig. 5. Impact of hyperparameters in the semi-supervised LS-SVM with the USPS dataset: we set μ equal to 0.1, 1, and 10 respectively, and vary β.

algorithms, the test error becomes high as the ratio N_l/N_u decreases, and the performance of the classifier is reduced dramatically. Third, we note that it is possible to build a powerful classifier with semi-supervised learning, which can be equivalent to a classifier obtained in the case of supervised learning. This is the goal we want to achieve with semi-supervised learning.

G. Relation between errors on unlabeled samples and classifier accuracy

In this section, we tested the classification errors on unlabeled samples in order to find the relation (correlation) between errors on the unlabeled samples and generalization errors (test

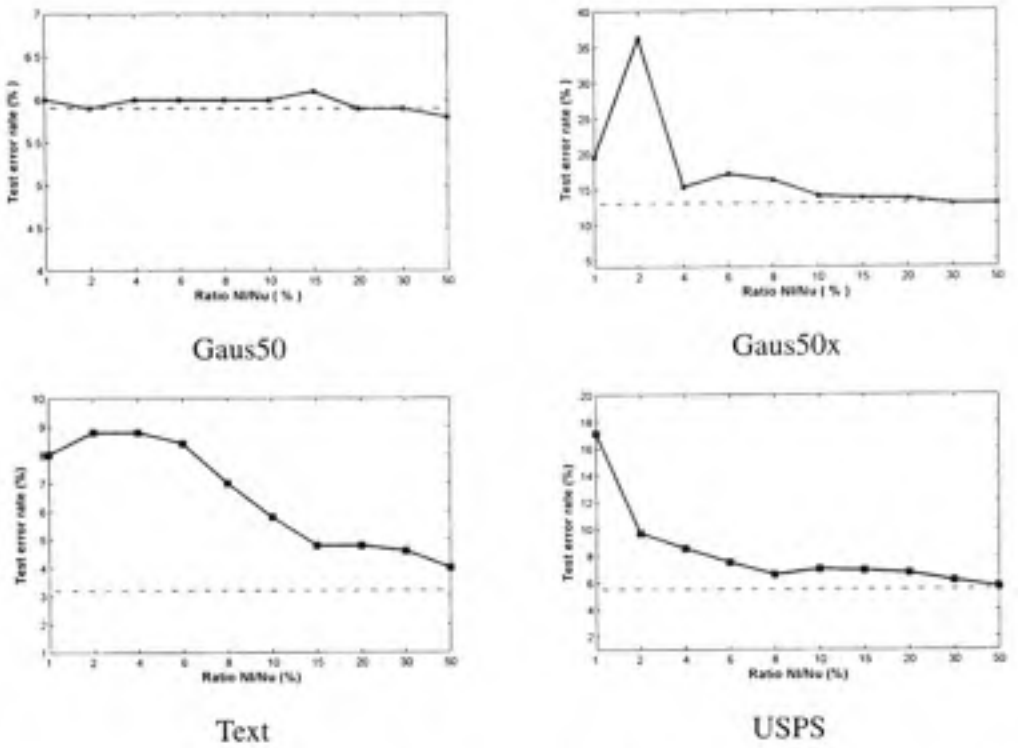


Fig. 6. The test error with respect to the ratio N_l/N_u is a solid line, where N_u represents the number of unlabeled samples and N_l the number of labeled samples. The dashed line corresponds to the error rate obtained on the same test set with $KPCA + Supervised - LS - SVM$ and all training data labeled.

error). We used the various semi-supervised problems generated in section VI-B with the Gauss50 dataset, the Gauss50x dataset, and the Text dataset. For each selection, after training the semi-supervised LS-SVM with our Bayesian two-level inference algorithm, we computed the error rate on the unlabeled samples (because we knew the true labels) and on the test set. The results are plotted in Figures 7, 8, and 9.

It is notable that the errors on the test set and on the unlabeled set are of the same order (Figures 7 and 8), meaning that the errors on the unlabeled training set are a reasonable estimate

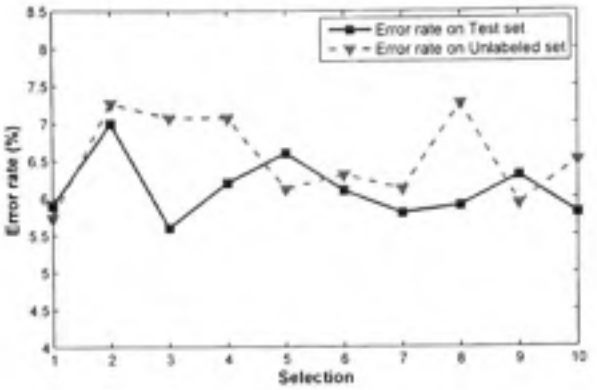


Fig. 7. Behavior of the error rate on the unlabeled samples and on the test samples with the Gaus50 dataset.

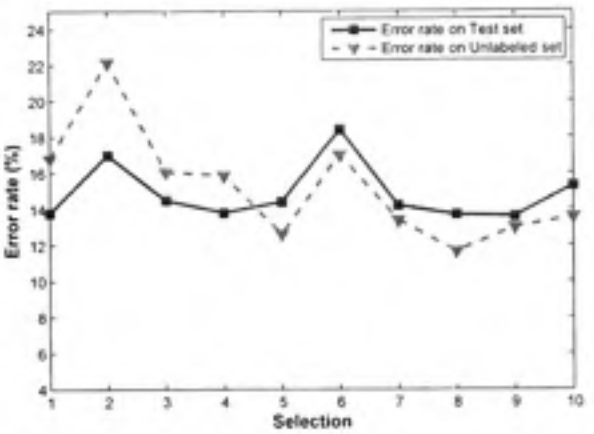


Fig. 8. Behavior of the error rate on the unlabeled samples and on the test samples with the Gaus50x dataset.

of the true performance of the system on unknown data. In addition, we note that there is no overfitting on the unlabeled data because the error rate on the unlabeled set is higher than the error rate on the test set, especially in the case of Text dataset (Figures 9). This demonstrates the robustness of our semi-supervised training algorithm based on the Bayesian technique. In short, the test error rate is either of the same order as the error rate on unlabeled data or lower than that rate.

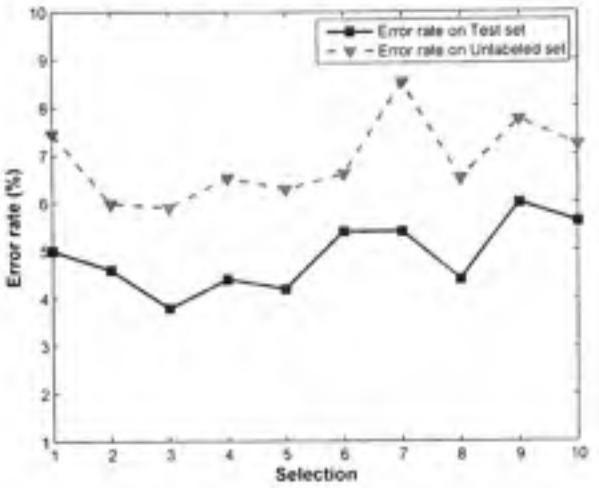


Fig. 9. Behavior of the error rate on the unlabeled samples and on the test samples with the Text dataset.

VII. Conclusion

In this paper, we take advantage of Bayesian analysis and hierarchical modeling to design a semi-supervised learning algorithm. Using this framework, we have established the Bayesian interpretation of the S^3VM introduced first time as a transductive SVM. We have also developed the semi-supervised training algorithm for the LS-SVM classifier based on two-level Bayesian inference with Gaussian process model. We have implemented and evaluated this method on both artificial and real data, and obtained encouraging results, which support the usefulness of our framework. We have shown that the performance of the semi-supervised LS-SVM compared with that of the semi-supervised SVM is promising.

In fact, the goal we wanted to achieve with semi-supervised learning is to build a powerful classifier which might perform as well as a classifier obtained in the supervised learning case. Thus, it will be interesting to study what ratio N_l/N_u is appropriate to achieve this goal. Although this ratio is dependent on the nature of the problem, a thorough study will help to give a more explanation and useful result about the upper bound of the generalization error.

We plan also in a future work, to use our framework to develop semi-supervised algorithms

for other classifiers and to establish the relation between our framework and other existing semi-supervised algorithms.

ACKNOWLEDGMENT

The authors would like to thank the NSERC of Canada for their financial support.

REFERENCES

[1] A.Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proc. 18th International Conf. on Machine Learning*, pages 19–26. Morgan Kaufmann, San Francisco, CA, 2001.

[2] Mathias M. Adankon and Mohamed Cheriet. Model selection for ls-svm. application to handwriting recognition. *Pattern Recognition. In press, doi:10.1016/j.patcog.2008.10.023.*

[3] Mathias M. Adankon and Mohamed Cheriet. Learning semi-supervised svm with genetic algorithm. In IEEE, editor, *International Joint Conference in Neural Networks 2007*, pages 1825 – 1830, Orlando, FL, 2007.

[4] Sundararajan S. Amrish Patel and Shirish Shevade. Semi-supervised classification using sparse gaussian process regression. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[5] Hyungil Ahn Ashish Kapoor, Yuan Qi and Rosalind Picard. Hyperparameter and kernel learning for graph based semi-supervised classification. In *Advances in Neural Information Processing Systems 18*, 2005.

[6] K. Bennett and A. Demiriz. Semi-supervised support vector machines, 1998.

[7] T. De Bie and N. Cristianini. Convex methods for transduction. In S. Thrun, L. Saul, and B. Scholkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2004.

[8] Alain Biem. Minimum classification error training for online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1041–1051, Jul. 2006.

[9] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK, 1995.

[10] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training.

In *Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 1998.

[11] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Computational Learing Theory*, pages 144–152, 1992.

[12] Alex Holub Carl Gold and Peter Sollich. Bayesian approach to feature selection and parameter tuning for support vector machine classifiers. *Neural Networks*, 18:693–701, 2005.

[13] V. Castelli and T. Cover. The relative value of labeled and unlabeled samples in pattern recognition with unknown mixing parameter. *IEEE Transactions on Information Theory*, 42:2101–2117, 1996.

[14] Gavin Cawley. Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In *proceedings IJCNN 2006, Vancouver, Canada*, July 2006.

[15] Gavin C. Cawley and Nicola L. C. Talbot. Improved sparse least-squares support vector machines. *Neurocomputing*, 48:1025–1031, 2002.

[16] Gavin C. Cawley and Nicola L. C. Talbot. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural Networks*, 17:1467–1475, 2004.

[17] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[18] O. Chapelle, V. Sindhwani, and S. S. Keerthi. Branch and bound for semi-supervised support vector machines. In *Proceedings of the NIPS 2006*, 2006.

[19] O. Chapelle, V. Sindhwani, and S.S. Keerthi. Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research*, 9:203–233, 2008.

[20] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

[21] W. Chu, C. J. Ong, and S. S. Keerthi. An improved conjugate gradient scheme to the solution of least squares svm. *IEEE Transactions on Neural Networks*, 16(2):498–501, 2005.

[22] Kok Seng Chua. Efficient computations for large least square support vector machine classifiers. *Pattern Recognition Letters*, 24:75–80, 2003.

[23] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Large scale transductive svm. *Journal of*

Machine Learning Research, 7:1687–1712, 2006.

[24] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[25] B. J. de Kruif and T. J. deVries. Pruning error minimization in least squares support vector machines. *IEEE Transactions on Neural Networks*, 14:696–702, 2003.

[26] G. Fung and O. Mangasarian. Semi-supervised support vector machines for unlabeled data classification. *Optimization Methods and Software*, 15:29–44, 2001.

[27] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 529–536. MIT Press, Cambridge, MA, 2004.

[28] CPLEX Optimization Incorporate. Using the cplex callable library and cplex mixed integer library. *Incline Village, Nevada*, 1995.

[29] Licheng Jiao, Liefeng Bo, and Ling Wang. Fast sparse approximation for least square support vector machine. *IEEE Transactions on Neural Networks*, 18(3):685–697, 2007.

[30] T. Joachims. Transductive learning via spectral graph partitioning. In *Proceeding of The Twentieth International Conference on Machine Learning (ICML-2003).*, 2003.

[31] Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

[32] S. Thrun K. Nigam, A. McCallum and T. Mitchell. Text classification from labeled and unlabeled documents using em. *machine Learning*, 39(2/3):103–134, 2000.

[33] N. D. Lawrence and M. Jordan. Semi-supervised learning via gaussian processes. In *Advances in Neural Information Processing Systems 17*, 2004.

[34] J. Luts, J.A.K. Suykens, and S. Van Huffel. Semi-supervised learning: avoiding zero label assumptions in kernel based classifiers. Technical report, Internal Report 07-122, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2007.

[35] J.I Marden. Hypothesis testing: from p values to bayes factors. *Journal of the American Statistical Association*, (95):131619, 2000.

[36] T. Mitchell. The role of unlabeled data in supervised learning. In *Proceedings of the Sixth International Colloquium on Cognitive Science, San Sebastian, Spain.*, 1999.

[37] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[38] Wiebe J. Riloff, E. and T. Wilson. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the Seventh Conference on Natural Language Learning*, 2003.

[39] Schneiderman H. Rosenberg C., Hebert M. Semi-supervised self-training of object detection models. In *WACV 2005.*, 2005.

[40] B. Scholkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[41] Matthias Seeger. Learning with labeled and unlabeled data. Technical report, Technical Report , Institute for Adaptive and neural Computation, University of Edinburgh, February 2001.

[42] Chu W. Sindhwani V. and Keerthi S. S. Semi-supervised gaussian process classifiers. In *Proceedings of international joint conferences on artificial intelligence*.

[43] Peter Sollich. Probabilistic methods for support vector machines. In *Conference on Advances in neural information processing systems 12*, pages 349–355. MIT Press, 2000.

[44] J.A.K. Suykens, L. Lukas, and L. Vandewalle. Sparse least squares support vector machine classifiers. In *European Symposium of Artificial Neural Networks*, 2000.

[45] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

[46] J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[47] Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[48] T. Van Gestel, J. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, and J. Vandewalle. Bayesian framework for least squares support vector machine classifiers, gaussian processes and kernel fisher discriminant analysis. *Neural Computation*, 15(5):1115–1148, 2002.

[49] T. Van Gestel, J.A.K. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and J. Vandewalle. Benchmarking least squares support vector machine classifiers. *Machine Learning*, 54(1):5–32, 2004.

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

[51] Song-Feng Zheng. Least square support vector machine and its bayesian interpretation.

Technical report, Technical Report, June 2004.

[52] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2007. http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf.

[53] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Semi-supervised learning: From gaussian fields to gaussian processes. Technical report, School of CS, CMU, 2003.

ARTICLE 4

HELP-TRAINING FOR SEMI-SUPERVISED LEARNING

M. M. Adankon[1], and M. Cheriet[1],
[1]École de Technologie Supérieure, 1100 Notre-Dame Ouest,
Montréal, Québec, Canada H3C 1K3

Cet article est soumis à Pattern Recognition.

Help-Training for Semi-supervised Learning

Mathias M. Adankon and Mohamed Cheriet

Synchromedia Laboratory for Multimedia Communication in Telepresence

ÉTS, 1100 Notre Dame-Ouest, Montréal, H3C 1K3, Canada

mathias.adankon@synchromedia.ca, mohamed.cheriet@etsmtl.ca

Abstract

In this paper, we propose to reinforce the Self-Training strategy in semi-supervised mode by using a generative classifier that may help to train the main discriminative classifier to label the unlabeled data. We call this semi-supervised strategy *Help-Training* and apply it to training kernel machine classifiers as support vector machines and as least-squares support vector machines. In addition, we propose a model selection strategy for semi-supervised training. Experimental results on both artificial and real problems demonstrate its usefulness relative to other classical semi-supervised methods.

Key words: Classification, semi-supervised learning, support vector machine, kernel machine.

1 Introduction

Pattern recognition problems are solved using classifiers, which are machines built using prototypes of the data to be recognized. Traditionally, machine learning needs labeled data, as does the training procedure called supervised learning. Now, labeling, as well as the data collection process itself, is a process which requires considerable human effort and is therefore very expensive. The labeling of handwritten

documents, images, or web pages, for example, requires both human expertise and insight, while in the field of medicine or biology, testing and very complex experiments are sometimes needed for this process. Moreover, in certain cases, it may be very difficult or even impossible to label all the available data.

Since we know that the larger the number of training samples, the better the performance of the classifier, the issue becomes one of finding a way to improve supervised learning by adding unlabeled data, or of determining what information can be obtained from unlabeled data to improve supervised learning [27]. Training using both labeled and unlabeled data is called semi-supervised learning [40, 32, 13]. In this type of training, the classifier is built by learning from both labeled and unlabeled data. Hence, it is not necessary to label all the data collected in order to train the classifier.

Self-Training is a classical technique used for semi-supervised learning [39] [1]. A classifier is first trained with the labeled data and the result is used to classify the unlabeled data. Thus, the unlabeled samples that are classified with the highest confidence score are added incrementally to the training set with their predicted labels and the process is then repeated until convergence is reached. However, this semi-supervised training algorithm does not give good results with discriminative classifiers. We consider here an example with a support vector machine (SVM) classifier in the Self-Training process (Fig. 1). This example shows what may happen if we use only the output of the SVM to identify unlabeled data that can be classified with a high level of confidence in order to add them to the training set with their predicted labels. In this case, a bad solution will be obtained at the end of the process, because the samples x_1 and x_2 will be classified in the wrong classes with high confidence scores. Thus, we have proposed using generative models to help discriminative classifiers make decisions during the self-labeling process in order

[1] This is the first well-known paper to have used the Self-Training strategy.

to overcome such misclassification problems. We have called this semi-supervised strategy *Help-Training* [3] which is a variant of Self-Training.

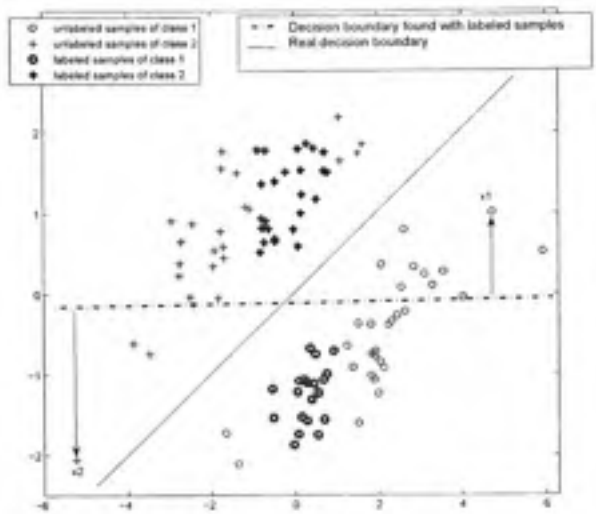


Fig. 1. Illustration of misclassification during Self-Training with a discriminative classifier like a SVM. Samples x1 and x2 will be classified in the wrong classes with a high level of confidence.

In this paper, we develop and apply the Help-Training method to train the semi-supervised SVM and its variant, the Least-Squares SVM (LS-SVMs), which are particular linear classifiers which are based on the margin-maximization principle. They use the kernel trick, a technique recently introduced within the machine learning community, to produce nonlinear boundaries. The idea behind kernels is to map training data nonlinearly into a higher-dimensional feature space via a mapping function Φ and to construct a separating hyperplane that maximizes the margin. The construction of the linear decision surface in this feature space only requires the evaluation of the dot product $\Phi(x).\Phi(y) = k(x, y)$, where $k()$ is called the kernel function [31].

SVMs perform structural risk minimization, which was introduced to machine learning by Vapnik [38], and they have yielded excellent generalization performance. Many applications have been developed successfully with SVMs. The idea

of using this classifier in a semi-supervised context is therefore an appealing one, and as a result, we can design a classifier for applications where a large amount of data is available without labeling all that data.

LS-SVMs have been proposed as a way to replace the convex quadratic programming problem with the convex linear system solving problem [34]. This is achieved by changing the hinge loss function in SVM by a squared-loss function. It has been shown through a meticulous empirical study that the generalization performance of the LS-SVM is comparable to that of the SVM [37]. In addition, the training algorithm of the LS-SVM is greatly simplified, since a linear problem is resolved instead of a quadratic programming (QP) problem in the SVM case. With this advantage, certain problems become much more tractable: for example model selection using the leave-one-out procedure [12, 10, 1]. While the LS-SVM has proven to be a successful alternative to standard SVMs, the number of applications in semi-supervised settings has been limited [26].

The rest of the paper is structured as follows: in section 2, we give a review of kernel hyperplane classification; in section 3, we describe a Help-Training algorithm; in sections 4 and 5, we describe an application of this algorithm to SVM and LS-SVM respectively; in section 6, we propose model selection for semi-supervised learning; and in section 7, we present the experimental results on both artificial and real data. We conclude the paper in section 8.

2 Kernel hyperplane classification

We consider kernel machines as the classifier of choice in this paper: in particular, the SVM and its variants.

Let us assume a binary classification problem and a training set $\mathcal{D}$ comprising ℓ labeled samples $\{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}$ and n unlabeled samples $\{x_1^\star, \ldots, x_n^\star\}$ form-

ing $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), ..., (x_\ell, y_\ell), x_1^*, x_2^*, ..., x_n^*\}$ with $x_i \in \mathbb{R}^d$, $x_i^* \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. The purpose of the classification scheme is to enable us to generate a map $f_\theta(x) = y : \mathbb{R}^d \to \{-1, 1\}$ from data to their labels. In machine learning, the complexity of the classification function f_θ greatly influences the performance achieved. Thus, in general, a highly complex function fits training data perfectly, but gives a poor generalization on unseen data [7]. In our case, we consider the classifiers based on a class of hyperplanes:

$$f_\theta(x) = sign[w'x + b] \tag{1}$$

where w' denotes the transpose of w, b is the bias term, and the parameter of the model $\theta = (w, b)$. This family of classifiers divides the input space into two parts: one for positive samples, the labels of which are equal to $y = 1$, and the second for negative samples with labels $y = -1$.

In real-world problems, the data are not linearly separable, and so, for many hyperplane classifiers, the kernel trick is used to produce nonlinear boundaries [9]. The idea behind kernels is to map training data nonlinearly into a higher-dimensional feature space via a mapping function Φ and to construct a separating hyperplane in this new space. The construction of the linear decision surface in this feature space only requires the evaluation of dot products $\phi(x_i).\phi(x_j) = k(x_i, x_j)$, where the application $k : \mathcal{R}^d \times \mathcal{R}^d \to \mathcal{R}$ is called the kernel function [31, 18]. Thus, the classification function becomes:

$$f_\theta(x) = sign[w'\phi(x) + b] = sign[\sum_i \alpha_i y_i k(x_i, x) + b] \tag{2}$$

where the $\alpha_i \in \mathbb{R}$ are scalars representing, with b, the classifier parameter in dual space.

2.1 *Review of the SVMs*

SVMs attempt to resolve the following optimization problem, which expresses the maximization of the margin $2/\|w\|$ and the minimization of the training error:

$$\min_{w,b,\xi} \frac{1}{2} w'w + C \sum_{i=1}^{\ell} \xi_i \tag{3}$$

$$\text{s.t.} : y_i[w'\phi(x) + b] \geq 1 - \xi_i \quad \forall i = 1, ..., \ell \tag{4}$$

$$\xi_i \geq 0 \quad \forall i = 1, ..., \ell \tag{5}$$

In its primal form, the Lagrangian of this problem is as follows:

$$\mathcal{T} = \frac{1}{2} w'w + C \sum_{i=1}^{\ell} \xi_i - \sum_{i=1}^{\ell} \alpha_i [y_i (w'\phi(x) + b) - 1 + \xi_i] - \sum_{i=1}^{\ell} \lambda_i \xi_i \tag{6}$$

with the Lagrange multipliers $\alpha_i \geq 0$ and $\lambda_i \geq 0$ for all $i = 1, ..., \ell$.

When we apply the differentiation theorem with respect to Lagrange, we obtain the classifier:

$$f(x) = sign[w'\phi(x) + b] = sign[\sum_{i=1}^{\ell} \alpha_i y_i k(x_i, x) + b] \tag{7}$$

with α the solution of (representing the dual problem):

$$maximize : W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{8}$$

$$s.t. : \sum_{i=1}^{\ell} \alpha_i y_i = 0 \quad and \quad 0 \leq \alpha_i \leq C, i = 1, ..., \ell$$

The threshold b is computed by averaging $b = y_j - \sum y_i \alpha_i k(x_i, x_j)$ over all support vectors x_j ($\alpha_j > 0$).

Semi-supervised SVMs are designed to find the hyperplane $< w, b >$ and the labels $y_1^*, \ldots, y_n^*$ of the unlabeled data that maximizes the margin with the minimum error, both on labeled and unlabeled data. Thus, we have:

$$\min_{w,b,\xi,\xi^*,y_1^*,\ldots,y_n^*} \frac{1}{2} w'w + C \sum_{i=1}^{\ell} \xi_i + C^* \sum_{j=1}^{n} \xi_j^* \tag{9}$$

$$\text{s.t.} : y_i[w'\phi(x_i) + b] \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i = 1, ..., \ell \tag{10}$$

$$y_j^*[w'\phi(x_j^*) + b] \geq 1 - \xi_j^*, \quad \xi_j^* \geq 0 \quad \forall j = 1, ..., n \tag{11}$$

The problem expressed in (9) is the semi-supervised SVM formulation used in the literature and many algorithms have been proposed to perform this optimization. This is a difficult problem because of the non-convex nature of the objective function. However, a number of solutions have been proposed with varying degrees of success, including an integer programming method [5], a combinatorial approach [24], and a sequential optimization procedure [21] with good scalable properties, albeit only demonstrated in the linear case. An interesting approach is given in [6], where the non-convex transductive problem is transformed into a convex, semi-definite programming problem. In general, the optimization problem is simplified with an appropriate choice of loss function. Further details on recent semi-supervised SVM optimization techniques can be found in [14].

2.2 *Review of the Least Squares SVMs*

Least-Squares SVMs (LS-SVMs) is a variant of the standard SVM. It is the result of answering the following question: "How much can the SVM formulation be simplified without losing any of its advantages?" Suykens and Vandewalle [34] proposed an LS-SVM in which the training algorithm solves a convex problem like

the SVM. Moreover, the training algorithm of the LS-SVM is greatly simplified, since a linear system problem is resolved instead of a quadratic programming (QP) problem. Also, it has been shown, through a meticulous empirical study, that the generalization performance of the LS-SVM is comparable to that of the SVM [37].

The LS-SVM is an interesting variant of the SVM; the standard Vapnik SVM classifier is modified to transform the quadratic programming problem into a linear system problem, yielding the optimization

$$\min_{w,b,\xi} \frac{1}{2} w'w + \frac{1}{2}\gamma \sum_{i=1}^{\ell} \xi_i^2 \tag{12}$$

$$\text{subject to } : \xi_i = y_i - [w'\phi(x_i) + b] \quad \forall i = 1, ..., \ell \tag{13}$$

which modifies the original SVM formulation at two points. First, the inequality constraints with the slack variable ξ_i expressed in (4) are replaced by equality constraints. Second, a squared loss function is considered in the objective function. These two essential modifications simplify the problem, which becomes a linear system.

The Lagrangian of problem (12) is expressed by :

$$\mathcal{S}(w, b, \xi, \alpha) = \frac{1}{2} w'w + \frac{1}{2}\gamma \sum_{i=1}^{\ell} \xi_i^2 - \sum_{i=1}^{\ell} \alpha_i \{y_i - [w'\phi(x) + b] - \xi_i\}$$

where α_i are Lagrange multipliers, which can be positive or negative because of equality constraints.

The conditions for optimality yield

$$\begin{cases} \frac{\partial \mathcal{S}}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{\ell} \alpha_i \phi(x_i) \\ \\ \frac{\partial \mathcal{S}}{\partial b} = 0 \Rightarrow \sum_{i=1}^{\ell} \alpha_i = 0 \\ \\ \frac{\partial \mathcal{S}}{\partial \xi_i} = 0 \Rightarrow \alpha_i = \gamma \xi_i, \quad \forall i = 1, ..., \ell \\ \\ \frac{\partial \mathcal{S}}{\partial \alpha_i} = 0 \Rightarrow \xi_i = y_i - [w' \phi(x_i) + b] \quad \forall i = 1, ..., \ell \end{cases}$$

We note that the system arising from these conditions (Karush-Kuhn-Tucker conditions) is linear, and that its solution is found by solving the system of linear equations expressed in the following matrix form:

$$\begin{pmatrix} K + \gamma^{-1} I & \vec{1}' \\ \\ \vec{1} & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \\ b \end{pmatrix} = \begin{pmatrix} Y \\ \\ 0 \end{pmatrix} \tag{14}$$

where $K_{ij} = k(x_i, x_j)$; $Y = (y_1, ..., y_\ell)'$; $\alpha = (\alpha_1, ..., \alpha_\ell)'$; and $\vec{1} = (1, ..., 1)$.

Several methods have been developed for supervised LS-SVMs, such as a training algorithm [34, 16, 15], model selection [12, 10, 1], and sparseness [33, 11, 19, 23]. However, to our knowledge, only the work of Juan et al. [26] on the semi-supervised LS-SVM is available. In this work, the authors introduced an extra variable and additional equality constraints into the classifier formulation in order to avoid the zero label assumption.

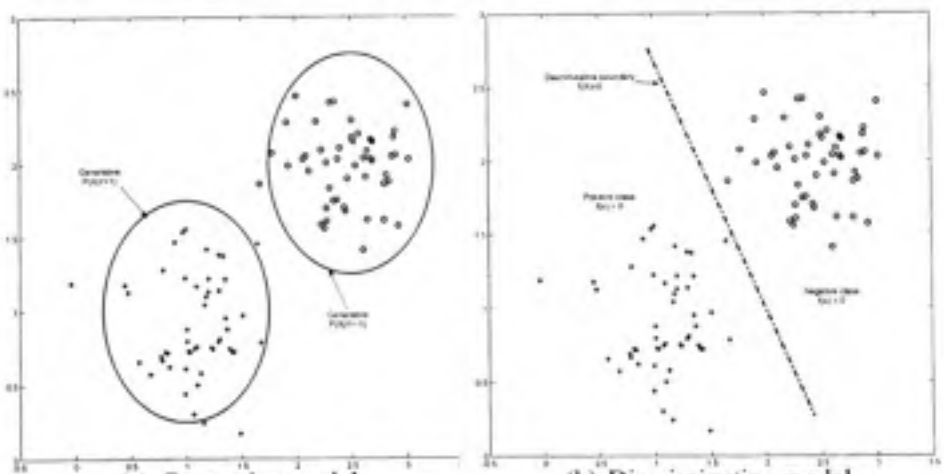
3 Help-Training algorithm

Building a classifier is equivalent to approximating an unknown target function $f : X \mapsto Y$ or estimating probability $P(Y|X)$, where X represents the inputs and Y the label.

One method consists of using the training data to learn estimates of $P(X|Y)$ and $P(Y)$ and predicting a new example by using the Bayes rules.

$$P(Y|X) = \frac{P(X|Y)P(Y)}{\sum_y P(X|y)P(y)} \tag{15}$$

The distribution $P(X|Y)$ can be viewed as describing how to generate X conditioned on the label Y [28]. Thus, this type of classifier is called a generative classifier. In contrast, the discriminative classifier is designed by modeling the posterior $P(Y|X)$ directly. For each new example, the label is computed by using $Y = f(X)$. These two ways to solve classification tasks are useful and complementary (see Figure 2), because each type of classifier possesses its own advantages. In Table 1, we summarize their advantages as reported in [36].



(a) Generative model (b) Discriminative model

Fig. 2. Illustration of generative versus discriminative classifier in a two-class problem.

Let us consider the main classifier C which is based on a discriminative approach,

Table 1
Advantages of discriminative vs generative approaches

Generative approaches	Discriminative approaches
They can deal with missing data or partially labeled data	They have better predictive performance (e.g. support vector machine).
They can readily handle compositionality (e.g. faces recognition with glasses and/or hats, and/or moustaches), whereas standard discriminative models need to see all combinations of possibilities during training.	They can be used to find accurate interclass boundaries, while the generative model is focused on each class distribution.
A new class can be added incrementally, because each class-conditional density $P(X\|Y)$ is learned inde-pendently.	They perform the prediction step (making predictions for new samples) very quickly.

and the classifier $\mathcal{G}$ which is based on a generative model. $\mathcal{G}$ produces a probability density model and attempts to find the basic formation of each class by modeling the data.

In Help-Training, $\mathcal{C}$ is helped by $\mathcal{G}$ to make a decision about which samples can be labeled and added to the training set since generative models return not just a classification but also a confidence level $P(Y|X)$. So, at each iteration, $\mathcal{G}$ is used to select the samples that have a high probability of belonging to each class. These selected samples constitute the candidate samples for the labeling process. After, $\mathcal{C}$ classifies the preselected samples and those that are classified with the most confident scores are added to the training set. The process is repeated until all unlabeled data are labeled. The details are provided in Algorithm 1.

The Help-Training method differs from the hybrid methods [25, 20, 8], in that, after

Algorithm 1 Outlines of the Help-Training algorithm

Input L= labeled samples, U= Unlabeled samples, classifiers $\mathcal{C}$ and $\mathcal{G}$.
Output Parameters of $\mathcal{C}$
Initialize the working set $W = L$
while $U \neq \phi$ **do**
- Train the classifiers $\mathcal{C}$ and $\mathcal{G}$ with W
- Select the samples which are classified with high probability from U with $\mathcal{G}$
- Compute the output of the classifier $\mathcal{C}$ for the selected samples
- Add the samples whose output are most confident to the working set W
- Remove the corresponding samples from the unlabeled set U

end while

training, only the discriminative classifier parameters are returned and used in the test step. Thus, the generative model is forgotten, since it is only used to help the main classifier $\mathcal{C}$ learn its parameters with unlabeled data.

4 Application to SVMs

In this section, we describe the application of the Help-Training algorithm to train semi-supervised SVMs. The main classifier (based on the discriminative approach) is the SVM. For the classifier based on the generative model, we chose a simple one using a non-parametric technique for probability density estimation. More specifically, we used the Parzen window estimator of the two class densities [29] which are defined by:

$$\mathcal{G}_+(x) = \frac{1}{\ell+} \sum_{i|y_i=+1} k_p(x_i, x) \tag{16}$$

$$\mathcal{G}_-(x) = \frac{1}{\ell-} \sum_{i|y_i=-1} k_p(x_i, x) \tag{17}$$

where $\ell+$ is the number of elements in the positive class, $\ell-$ the number of negative samples, and k_p the Parzen kernel function.

Given a set of data, the probability distribution that generated the data can be estimated in two ways. First, the functional form of the probability density function is specified in advance and we use a given dataset to estimate the parameters of that function. This method is referred to as parametric. Second, we have non-parametric method, where the probability distribution is modeled without making any assumption about the functional form of the probability density function.

The Parzen window method, or kernel density estimation, is a non-parametric technique for probability density estimation. Given a set of data, we estimate the probability distribution that generates the data using a linear combination of kernels centered on the observed points. In other words, the Parzen window method gives the probability distribution as the average of kernel functions with each data point as a center.

Algorithm 2 shows the details of the application of Help-Training to semi-supervised SVMs where the Parzen window estimator is used to help the SVM classifier label the unlabeled samples in training set.

5 Application to LS-SVMs

The process of Help-Training semi-supervised LS-SVMs is similar to that of SVMs, the difference being that with LS-SVMs, it is possible to derive incremental learning.

Training LS-SVMs is equivalent to solving the linear system (14). This leads to inversion of the main matrix in this equation and computation of the parameter

Algorithm 2 Help-Training algorithm for semi-supervised SVM

Input L= labeled samples, U= Unlabeled samples.
Output Parameters of SVM
Initialize the working set $W = L$ and $\epsilon = \epsilon_0$
while $U \neq \phi$ **do**
- Train the SVM with the working set W
- Estimate the probability density model $\mathcal{G}_+$ for positive samples in W
- Estimate the probability density model $\mathcal{G}_-$ for negative samples in W
- Select n_1 samples from U with high probability according to $\mathcal{G}_+$
- Select n_2 samples from U with high probability according to $\mathcal{G}_-$
- Compute the output of the $LS - SVM$ for the selected $(n_1 + n_2)$ samples
- Constitute the set S formed by the samples whose output are most confident, $f(x) \geq \epsilon$
- Update the working set $W \leftarrow W \cup S$
- Update the unlabeled set $U \leftarrow U - S$
- Reduce the value of ϵ if $S = \phi$

end while
- Retrain the SVM
- Return the final parameters of the SVM.

$\theta = (b, \alpha)$ of the classifier. Thus, we can write:

$$\begin{pmatrix} b \\ \\ \alpha \end{pmatrix} = \begin{pmatrix} 0 & \vec{1}' \\ \\ \vec{1} & K + \gamma^{-1} I \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ \\ Y \end{pmatrix} \tag{18}$$

Considering algorithm 2, at each iteration t, the classifier is trained with a small amount of additional data, the set S selected from unlabeled data during the previous iteration $t - 1$. Thus, the inverse of the new extended matrix can be computed using the block matrix inversion lemma. Making an assumption at iteration t, the working set $W = \{(x_1, y_1), \ldots, (x_\ell, y_\ell), (x_1^*, y_1^*), \ldots, (x_m^*, y_m^*)\}$ with ℓ labeled samples at the beginning of the training process and m unlabeled samples labeled during the previous iterations $1, ..., t-1$. At the end of the iteration t, s samples are added to the working set, which becomes

$W = \{(x_1, y_1), \ldots, (x_\ell, y_\ell), (x_1^*, y_1^*), \ldots, (x_m^*, y_m^*), (x_{m+1}^*, y_{m+1}^*), \ldots, (x_{m+s}^*, y_{m+s}^*)\}$

for the next iteration $t + 1$.

Then, we have, at iteration t,

$$\theta_t = Q_t^{-1}\widetilde{Y}_t \tag{19}$$

with

$$Q_t = \begin{pmatrix} 0 & 1 & \dots & 1 & 1 & \dots & 1 \\ 1 & k(x_1,x_1)+\gamma^{-1} & \dots & k(x_1,x_\ell) & k(x_1,x_1^*) & \dots & k(x_1,x_m^*) \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & k(x_\ell,x_1) & \dots & k(x_\ell,x_\ell)+\gamma^{-1} & k(x_\ell,x_1^*) & \dots & k(x_\ell,x_m^*) \\ 1 & k(x_1^*,x_1) & \dots & k(x_1^*,x_\ell) & k(x_1^*,x_1^*)+\gamma^{-1} & \dots & k(x_1^*,x_m^*) \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & k(x_m^*,x_1) & \dots & k(x_m^*,x_\ell) & k(x_m^*,x_1^*) & \dots & k(x_m^*,x_m^*)+\gamma^{-1} \end{pmatrix} \tag{20}$$

and

$$\widetilde{Y}_t = (0, y_1, \dots, y_\ell, y_1^*, \dots, y_m^*)' . \tag{21}$$

At iteration $t+1$,

$$\theta_{t+1} = Q_{t+1}^{-1}\widetilde{Y}_{t+1} \tag{22}$$

with

$$Q_{t+1} =$$

$$\begin{pmatrix}
0 & 1 & \dots & 1 & 1 & \dots & 1 \\
1 & k(x_1,x_1)+\gamma^{-1} & \dots & k(x_1,x_\ell) & k(x_1,x_1^*) & \dots & k(x_1,x_{m+s}^*) \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots \\
1 & k(x_\ell,x_1) & \dots & k(x_\ell,x_\ell)+\gamma^{-1} & k(x_\ell,x_1^*) & \dots & k(x_\ell,x_{m+s}^*) \\
1 & k(x_1^*,x_1) & \dots & k(x_1^*,x_\ell) & k(x_1^*,x_1^*)+\gamma^{-1} & \dots & k(x_1^*,x_{m+s}^*) \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots \\
1 & k(x_{m+s}^*,x_1) & \dots & k(x_{m+s}^*,x_\ell) & k(x_{m+s}^*,x_1^*) & \dots & k(x_{m+s}^*,x_{m+s}^*)+\gamma^{-1}
\end{pmatrix} \tag{23}$$

and

$$\widetilde{Y}_{t+1} = \left(0, y_1, \dots, y_\ell, y_1^*, \dots, y_{m+s}^*\right)'. \tag{24}$$

The matrix Q_{t+1} can be partitioned into 4 blocks:

$$Q_{t+1} = \begin{pmatrix} Q_t & K_{s1} \\ K_{s1}' & K_{ss} \end{pmatrix} \tag{25}$$

with

$$K_{s1} = \begin{pmatrix} 1 & \dots & 1 \\ k(x_1, x^*_{m+1}) & \dots & k(x_1, x^*_{m+s}) \\ \dots & \dots & \dots \\ k(x_\ell, x^*_{m+1}) & \dots & k(x_\ell, x^*_{m+s}) \\ k(x^*_1, x^*_{m+1}) & \dots & k(x^*_1, x^*_{m+s}) \\ \dots & \dots & \dots \\ k(x^*_m, x^*_{m+1}) & \dots & k(x^*_m, x^*_{m+s}) \end{pmatrix} \tag{26}$$

and

$$K_{ss} = \begin{pmatrix} k(x^*_{m+1}, x^*_{m+1}) + \gamma^{-1} & \dots & k(x^*_{m+1}, x^*_{m+s}) \\ \dots & \dots & \dots \\ k(x^*_{m+s}, x^*_{m+1}) & \dots & k(x^*_{m+s}, x^*_{m+s}) + \gamma^{-1} \end{pmatrix} \tag{27}$$

Using the partitioned matrix lemma, the inverse can then be written as:

$$Q^{-1}_{t+1} = \begin{pmatrix} Q^{-1}_t + Q^{-1}_t K_{s1} D'_s & -D_s \\ -D'_s & C_s \end{pmatrix} \tag{28}$$

with

$$C_s = \left[K_{ss} - K'_{s1} Q^{-1}_t K_{s1} \right]^{-1} \tag{29}$$

$$D_s = Q^{-1}_t K_{s1} C_s \tag{30}$$

Thus, with equation (28), we can update the LS-SVM classifier parameters saving computational time at each iteration, because the computational complexity of the inverse Q_{t+1}^{-1} is reduced by $\mathcal{O}((\ell + m + 1)^3 + s(\ell + m + 1)^2 + s^2(\ell + m + 1))$ operations [2].

6 Model Selection

Model selection for the SVM/LS-SVM consists of selecting the hyperparameters that yield the best performance of the machine. The SVM/LS-SVM classifiers have two types of hyperparameter: the regularization parameter C for SVM, and γ for LS-SVM, which controls the trade-off between training error minimization and margin maximization, and the kernel parameters that define the given kernel function.

6.1 Cross-validation

The classifier is designed to correctly classify unseen objects which are not used during the training process. Generalization is the capacity of the classifier to respond to this task. When a classifier has a good generalization capacity, it can correctly classify unseen examples. The cross-validation procedure is a good technique for evaluating classifier generalization performance, the idea being to test the generalization capacity of the classifier through unseen data.

In k-fold cross-validation, we divide the available training data into k subsets. We train the machine k times, each time leaving out one of the subsets from training,

[2] This evaluation is performed by considering the product of two matrices, the sizes of which are $m \times p$ and $p \times n$, requires $\mathcal{O}(mnp)$ operations, while the inverse of the squared matrix needs $\mathcal{O}(n^3)$ operations.

and use only the omitted subset to compute the given error criterion. The average error rate obtained over the k operations, gives an estimation of the classifier generalization capacity. The variance of the result is reduced when k is increased. If k equals the size of the training set, the maximum value, this is called "leave-one-out" cross-validation, or LOO cross-validation.

6.2 Empirical Error criterion

Several criteria have been developed to perform model selection for kernel machines, in particular for SVMs. In this paper, we chose to use the empirical error criterion developed in [4, 2].

Let us define $t_i = (y_i + 1)/2$. The empirical error is given by the following expression:

$$E_i = |t_i - \hat{p}_i| \tag{31}$$

where $\hat{p}_i$ is the estimated posterior probability corresponding to the data example x_i.

The estimated posterior probability is determined by :

$$\hat{p}_i = \frac{1}{1 + exp(A.f_i + B)} \tag{32}$$

where $f_i = f(x_i)$ and the parameters A and B are fitted after minimizing the cross-entropy error [7] as proposed by Platt in [30]. In this paper, we use fixed values for A and B, because we need to keep the continuity of the empirical error expression for each validation sample at each iteration.

We assume that the kernel function depends on one or several parameters, encoded within the vector $\lambda = (\lambda_1, \lambda_2, ...)$ including the hyperparameter γ for LS-SVM and C for SVM. These parameters are optimized by minimizing the empirical error $E = \sum E_i$. However, sometimes our problem is not convex, and, to overcome this situation, we can use many starting points. We can also use the simple function *fminsearch* implemented in Matlab with different starting points.

In practice, for SVMs, the empirical error is minimized on the validation set [4, 2]. But, in the semi-supervised context, only a small number of the labeled samples are available. Thus, it is inappropriate to reduce this set to form the validation set. We propose to apply cross-validation procedure only on labeled data. We divide the labeled set into k subsets and use $k-1$ subsets with all unlabeled data for performing the training task at each iteration. Then, the empirical error is estimated only on the omitted labeled subset.

$$\lambda_{optimal} = \underset{\lambda}{\operatorname{argmin}} \sum_{t=1}^{k} \sum_{(x_i,y_i)\in\mathcal{V}_t} \left| \frac{y_i + 1}{2} - \frac{1}{1 + exp[A.f^{(-\mathcal{V}_t)}(x_i) + B]} \right| \quad (33)$$

where:

$\mathcal{V}_t$ is the omitted labeled subset,

$\mathcal{D}_t = \mathcal{V}_1 \cup \mathcal{V}_2 \ldots \mathcal{V}_k$ represents the set of the labeled data, and

$f^{(-\mathcal{V}_t)}$ is the classifier trained without the subset $\mathcal{V}_t$.

Concerning LS-SVMs, we use the LOO procedure on labeled data using our pre-

vious work [1]. In this case, $\mathcal{V}_t = \{(x_i, y_i)\}$ and the LOO prediction for the ith labeled sample is given by:

$$f^{(-i)}(x_i) = y_i - \frac{\alpha_i}{H_{ii}^{-1}} \tag{34}$$

with

$$H = \begin{pmatrix} K + \gamma^{-1} I & \vec{1}' \\ \vec{1} & 0 \end{pmatrix} \tag{35}$$

7 Experiments

7.1 Datasets

7.1.1 Gaus50

We consider a two-class problem in a 50-dimensional input space like the artificial datasets used in [22]. The data of each class are generated with equal probability from a Gaussian distribution with a unit covariance matrix. The mean of the Gaussian is $(a, a, ..., a)$ for class 1 and $-(a, a, ..., a)$ for class 2. In order to obtain a Bayes error equal to 5%, the parameter a is fixed to 0.23. We generated 550 samples for training and 1000 samples for testing.

7.1.2 Gaus50x

We have here another artificial two-class problem in a 50-D space but with multimodal distribution. The data in each class are generated with equal probability from a Gaussian mixture distribution with significant overlap. The two classes have

equal priors and their class-conditional distributions are, respectively, $p(x|y=1) = 0.49\mathcal{N}(\mu_1, I) + 0.51\mathcal{N}(\mu_2, I)$ and $p(x|y=-1) = 0.49\mathcal{N}(-\mu_1, I) + 0.51\mathcal{N}(-\mu_2, I)$ where $\mu_1 = (0.25, 0.25, ..., 0.25, 0.25)$, $\mu_2 = (0.25, 0.25, ..., -0.25, -0.25)$, $N(\mu, I)$ is a Gaussian density function with mean vector μ, and I is the identity matrix for its covariance matrix. We generated 550 samples for training and 1000 samples for testing.

7.1.3 Text

Text document classification is a problem involving high dimensionality. We used the *Mac* and *Windows* classes taken from the 20 news group dataset, with 7511 dimensions, as preprocessed in [35]. We divided the dataset into two subsets: 1446 samples for training and 500 for testing.

7.1.4 USPS

USPS is the well-known US Postal Service handwritten digits recognition corpus. The digits are represented by normalized grey scale images of size 16×16. The learning dataset contains 7291 samples for training, while the testing dataset consists of 2007 other samples. We have a multi-class problem with 10 classes. We then trained 10 machines using the *one-against-all* strategy.

7.2 Testing our model selection technique

First, we consider that the SVM/LS-SVM is trained on the entire dataset with the real labels, i.e. supervised learning. The hyperparameters are selected by cross-validation in this training mode. Then, we use these values in semi-supervised learning as optimal values of the Gaussian kernel width and hyperparameters γ/C,

referring to this method as *baseline*. Note that in the real world this method is not realizable, because we do not have access to the real labels for the entire dataset in semi-supervised training. We therefore only use it for comparison purposes.

For the sake of simplicity, we used the same SVM/LS-SVM kernel function as the Parzen kernel function.

$$\mathcal{G}_+(x) = \frac{1}{\ell+} \sum_{i|y_i=+1} exp(-\frac{\|x_i - x\|^2}{\sigma^2}) \tag{36}$$

$$\mathcal{G}_-(x) = \frac{1}{\ell-} \sum_{i|y_i=-1} exp(-\frac{\|x_i - x\|^2}{\sigma^2}) \tag{37}$$

where σ is the kernel parameter.

Second, we perform model selection in semi-supervised learning mode. We select the hyperparameter values by minimizing the empirical error on labeled data combined with unlabeled data, using equation (33). We randomly selected a part of the training set to form the labeled data, the remaining of data consisting of the unlabeled samples. We repeated this operation 10 times. We test the classifiers on the same test set and we report the results in tables 2 and 3. For the Gaus50 dataset, we select 50 labeled samples each time and for the Text dataset 144 samples are selected.

We note that our model selection strategy achieved a good result, as the generalization errors obtained are approximatively identical with the baseline ones. In particular, in the case of the LS-SVM with the Text dataset, our model selection strategy outperforms the baseline method, thereby confirming our approach. Thus, it is possible to perform model selection with a small amount of labeled data using our strategy.

Table 2
Comparison between our model selection technique and the baseline (Test error rate obtained on the Gaus50 dataset)

Selection	Test error rate in percent with SVM		Test error rate in percent with LS-SVM	
	Baseline	Our model selection technique	Baseline	Our model selection technique
#1	6.3	6.5	5.3	5.6
#2	7.0	7.3	5.8	5.6
#3	5.9	6.4	5.9	6.2
#4	6.4	6.6	6.2	5.7
#5	6.7	6.4	6.2	5.9
#6	6.2	6.7	6.1	6.4
#7	6.0	6.2	6.4	6.3
#8	6.3	6.7	5.5	6.0
#9	6.9	8.3	5.6	5.7
#10	6.0	6.2	5.7	5.9
Mean	6.37	6.73	5.87	5.93

7.3 Comparison with Self-Training

To quantify the effect of semi-supervised learning and our Help-Training strategy, we ran a set of experiments using the SVM and LS-SVM classifiers trained with either labeled data or a mix of labeled and unlabeled data. We randomly selected 10% of the training set to form the labeled data, the remaining 90% of data consisting of the unlabeled samples. We repeated this operation 10 times, each time testing the semi-supervised SVM/LS-SVM with the Self-Training and the Help-Training method, and the single SVM/LS-SVM trained only with labeled data. The RBF kernel is used with $\gamma = 1/\sigma^2 = 0.02$ and the hyperparameters C is set to 1. The results are shown in Table 4, 5, 6 and 7. We have three main comments on those results.

Table 3
Comparison between our model selection technique and the baseline (Test error rate obtained on the Text dataset)

Selection	Test error rate in percent with SVM		Test error rate in percent with LS-SVM	
	Baseline	Our model selection technique	Baseline	Our model selection technique
#1	3.8	4.6	4.8	4.2
#2	3.8	3.8	4.6	4.0
#3	4.2	4.2	4.8	3.6
#4	3.2	3.0	4.8	2.8
#5	4.2	4.6	5.0	4.6
#6	4.0	4.0	3.4	4.2
#7	4.0	4.0	4.2	3.8
#8	2.6	3.4	5.8	3.4
#9	3.6	2.8	5.0	3.0
#10	4.2	4.6	6.2	4.4
Mean	3.76	3.90	4.86	3.80

First, both semi-supervised learning methods, Self-Training and Help-Training, give good results. Adding unlabeled data improves the generalization capacity of the classifier, and sometimes we approach the Bayes error. For example, the Bayes error for the artificial problem is 5% and the semi-supervised LS-SVM with Help-Training gives a 5.3% error rate for selection #1.

Second, Self-Training SVM/LS-SVM performance depends on the initial distribution information given by the labeled data. Thus, the generalization capacity of the Self-Training SVM/LS-SVM varies with respect to the generalization capacity of supervised learning with labeled data (see Fig. 3). However, the generalization performance of the Help-Training SVM/LS-SVM is less dependent on the initial distribution information.

Third, our method, Help-Training, performs better than the Self-Training strategy. It is also remarkable that the performance of the classifiers trained with Help-Training have nearly the same performance as the classifier trained with all samples in supervised mode (5.7% error rate on test set).

Table 4
Comparison between Self-Training and Help-Training: Error rate (%) on the Gaus50 dataset with the SVM classifier

Selection	Supervised SVM with labeled data	Semi-supervised SVM with Self-Training	Semi-supervised SVM with Help-Training
#1	10.9	6.7	6.3
#2	15.2	7.0	7.0
#3	10.1	6.4	5.9
#4	8.7	6.1	6.4
#5	11.4	6.3	6.7
#6	10.0	5.7	6.2
#7	11.5	6.9	6.0
#8	10.7	7.7	6.3
#9	9.1	5.7	6.9
#10	11.9	5.6	6.0
Mean	10.95	6.41	6.37

7.4 Impact of the ratio of the labeled samples

We also tested the behavior of our algorithm with respect to the number N_l of labeled samples. We compare the results with those found with the supervised SVM/LS-SVM and the semi-supervised LS-SVM with Self-Training. Figures 4,

Table 5
Comparison between Self-Training and Help-Training: Error rate (%) on the Gaus50 dataset with the LS-SVM classifier

Selection	Supervised LS-SVM with labeled data	Semi-supervised LS-SVM with Self-Training	Semi-supervised LS-SVM with Help-Training
#1	8.7	6.1	5.3
#2	23.1	23.7	5.8
#3	12.9	8.8	5.9
#4	8.7	6.3	6.2
#5	14.6	8.5	6.2
#6	9.3	6.2	6.1
#7	16.2	7.2	6.4
#8	15.9	11.5	5.5
#9	9.4	6.2	5.6
#10	7.4	6.2	5.7
Mean	12.62	9.07	5.87

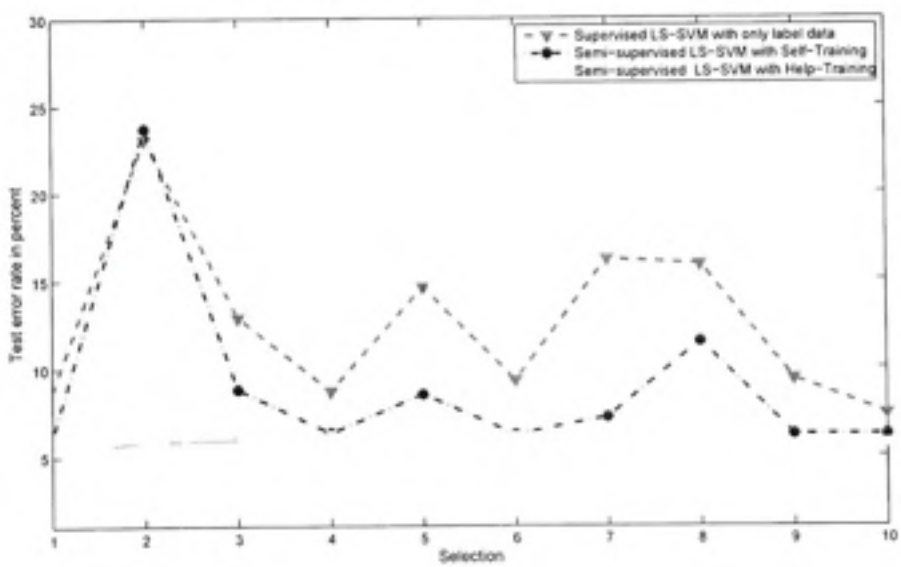


Fig. 3. Correlation between supervised and semi-supervised performance: test error rate obtained on the Gaus50 dataset with respect to each selection.

Table 6
Comparison between Self-Training and Help-Training: Error rate (%) on the Text dataset with the SVM classifier

Selection	Supervised SVM with labeled data	Semi-supervised SVM with Self-Training	Semi-supervised SVM with Help-Training
# 1	8.2	3.4	3.8
# 2	6.6	3.8	3.8
# 3	50.4	3.4	4.2
# 4	50.4	3.2	3.2
# 5	6.0	4.0	4.2
# 6	50.4	50.4	4.0
# 7	9.2	4.4	4.0
# 8	50.4	50.4	2.6
# 9	7.6	35.0	3.6
# 10	50.4	49.8	4.2
Mean	28.96	20.78	3.76

5, 6 and 7 show the variation of the test error when we increased the number of labeled samples in the training data. We see that the test error for the supervised SVM/LS-SVM and the semi-supervised SVM/LS-SVM with Self-Training drops as the number N_l of labeled data increases. Since we have more information about the sample labels, the generalization error is improved.

However, we note that the test error rate obtained with the semi-supervised SVM/LS-SVM with Help-Training is good, and less dependent on the number of labeled samples in the training set. Thus, using the generative model is very useful and makes the resulting classifier more robust.

Table 7
Comparison between Self-Training and Help-Training: Error rate (%) on the Text dataset with the LS-SVM classifier

Selection	Supervised LS-SVM with labeled data	Semi-supervised LS-SVM with Self-Training	Semi-supervised LS-SVM with Help-Training
# 1	10.8	8.8	4.8
# 2	48.2	13.8	4.6
# 3	48.6	11.0	4.8
# 4	50.4	20.2	4.8
# 5	45.0	14.4	5.0
# 6	49.6	4.4	3.4
# 7	49.6	5.0	4.2
# 8	34.2	11.2	5.8
# 9	15.0	9.4	5.0
# 10	47.8	5.6	6.2
Mean	39.92	10.38	4.86

7.5 *Comparison of our semi-supervised LS-SVM with other semi-supervised algorithms*

For the sake of comparison, we compared our algorithm with the semi-supervised SVM (S^3VM) methods [14], the code of which is available on the Web. The following S^3VM algorithms were tested:

- $S^3VM - light$, which uses a combinatorial search over the unlabeled data while keeping the ratio of positive versus negative the same as the ratio in the labeled set [24];
- $S^3VM - CCCP$ (concave-convex procedure), which decomposes a non-convex loss function into a convex part and a concave part, and where this last part is iteratively approximated by a Taylor expansion [17].

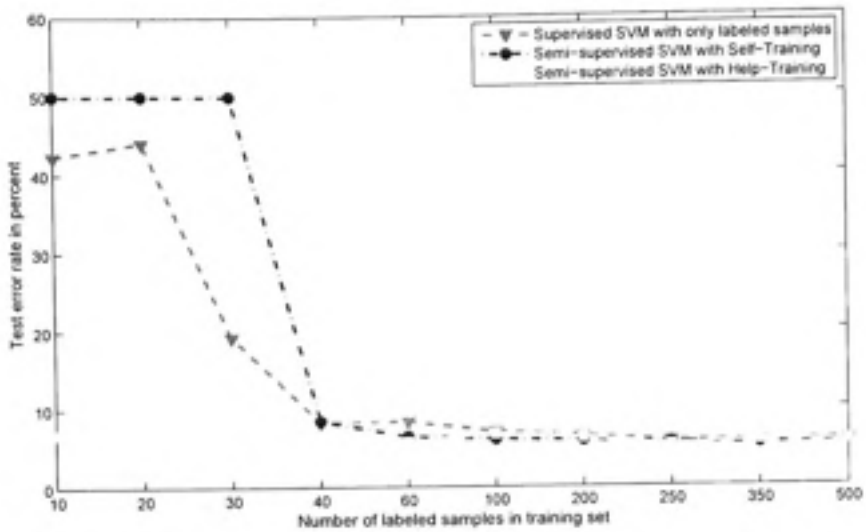


Fig. 4. Behavior of SVM algorithms with respect to the number of labeled samples on the Gaus50 dataset. Plot of test error versus the number of labeled samples.

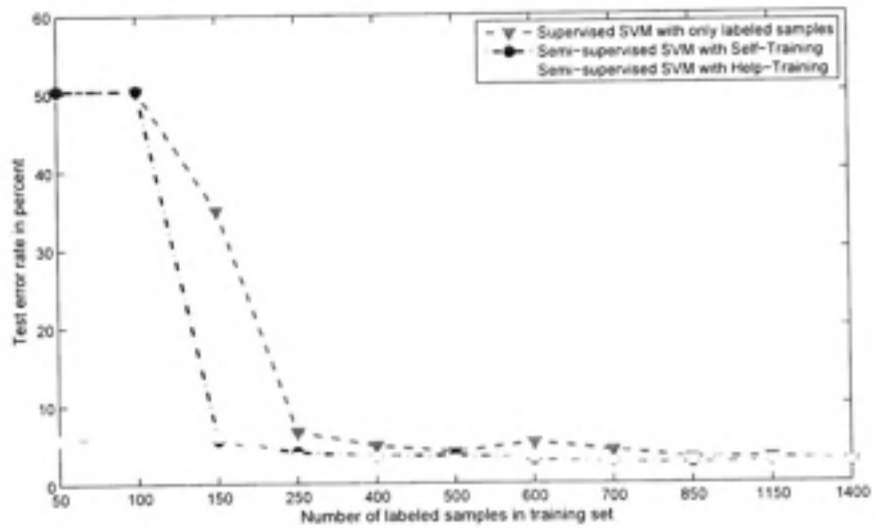


Fig. 5. Behavior of SVM algorithms with respect to the number of labeled samples on the Text dataset. Plot of test error versus the number of labeled samples.

We used both the artificial and the real datasets described below. For each dataset, we labeled 10% of the training samples and kept the rest as unlabeled data. The error rate is computed on the test set. The results are reported in Table 8. We note

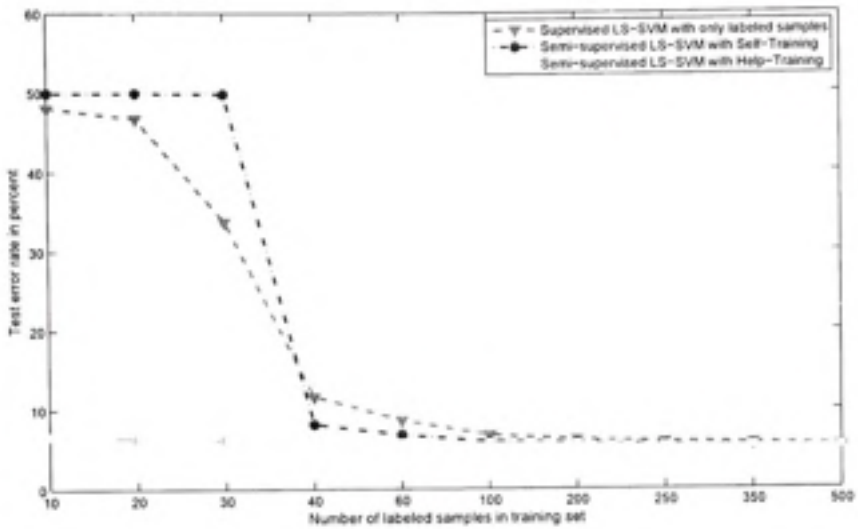


Fig. 6. Behavior of LS-SVM algorithms with respect to the number of labeled samples on the Gaus50 dataset. Plot of test error versus the number of labeled samples.

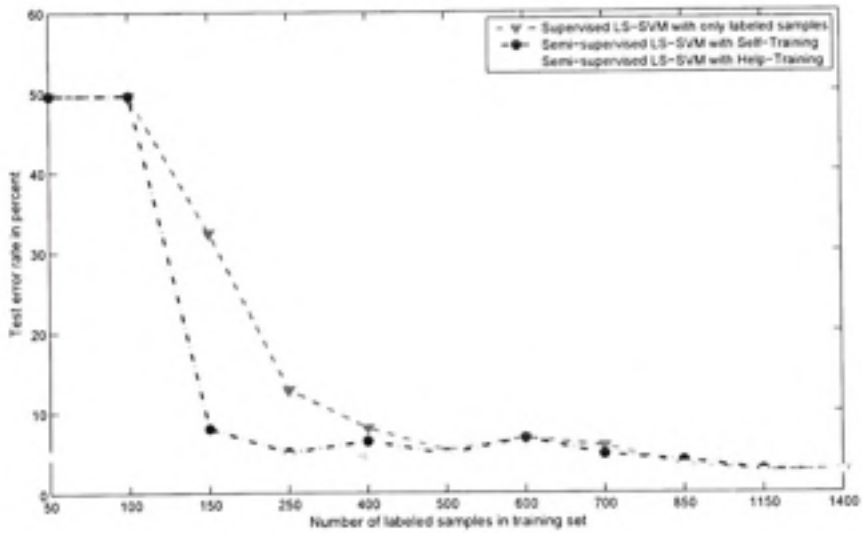


Fig. 7. Behavior of LS-SVM algorithms with respect to the number of labeled samples on the Text dataset. Plot of test error versus the number of labeled samples.

that our semi-supervised SVM/LS-SVM algorithm with Help-Training achieved a good performance, and that its generalization performance is comparable to that of the semi-supervised SVM.

Table 8
Comparison between Semi-supervised SVM and Semi-supervised LS-SVM: $S^3VM - light$ [24] and $S^3VM - CCCP$ [17] train the semi-supervised SVM while $HT - SVM$ and $HT - LS - SVM$ train the semi-supervised SVM/LS-SVM with our Help-Training strategy. In the table we report the error rate on the test set.

	$S^3VM - light$	$S^3VM - CCCP$	$HT - SVM$	$HT - LS - SVM$
Gaus50	8.80%	7.70%	6.30%	**5.9%**
Gaus50x	16.30%	**15.50 %**	17.50%	15.80%
Text	5.40%	5.45%	**4.20%**	**4.20%**
USPS	14.70%	7.32%	7.77%	**6.02%**

8 Conclusions

We first developed Help-Training for the semi-supervised SVM [3]. In this paper, we present the expanded version and extended our idea to the semi-supervised LS-SVM. Moreover, we proposed model selection for the semi-supervised context, which yielded interesting results. We tested our algorithm on artificial data as well as on real data, and showed that our algorithm performs better than the Self-Training algorithm. Also, compared with $S^3VM - light$, the first implementation of the semi-supervised SVM, and $S^3VM - CCCP$, which used the concave-convex procedure, our method achieved a good performance.

In future work, we plan to use the kernel machines with generative models other than the Parzen windows estimator, in order to improve the training process. It will also be interesting to study the impact of using different kernel functions for the kernel classifier and the Parzen windows estimator.

Acknowledgment

The authors would like to thank the NSERC of Canada for their financial support.

References

[1] Mathias M. Adankon and Mohamed Cheriet. Model selection for ls-svm. application to handwriting recognition. *Pattern Recognition, In press*.

[2] Mathias M. Adankon and Mohamed Cheriet. Optimizing resources in model selection for support vector machines. *Pattern Recognition*, 40(3):953–963, 2007.

[3] Mathias M. Adankon and Mohamed Cheriet. Help-training for semi-supervised discrimininative classifier. application to svm. In *19th Internatioonale Conference on Pattern Recognition*, Tampa, USA, 2008.

[4] N. E. Ayat, M. Cheriet, and C.Y. Suen. Automatic model selection for the optimization of the svm kernels. *Pattern Recognition*, 38(10):1733–1745, 2005.

[5] K. Bennett and A. Demiriz. Semi-supervised support vector machines, 1998.

[6] T. De Bie and N. Cristianini. Convex methods for transduction. In S. Thrun, L. Saul, and B. Scholkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2004.

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

[8] A. Bosch, A. Zisserman, and X. Muoz. Scene classification using a hybrid generative/discriminative approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(4):712–727, 2008.

[9] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Computational Learing Theory*, pages 144–152, 1992.

[10] Gavin Cawley. Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In *proceedings IJCNN 2006, Vancouver, Canada*, July 2006.

[11] Gavin C. Cawley and Nicola L. C. Talbot. Improved sparse least-squares support vector machines. *Neurocomputing*, 48:1025–1031, 2002.

[12] Gavin C. Cawley and Nicola L. C. Talbot. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural Networks*, 17:1467–1475, 2004.

[13] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[14] O. Chapelle, V. Sindhwani, and S.S. Keerthi. Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research*, 9:203–233, 2008.

[15] W. Chu, C. J. Ong, and S. S. Keerthi. An improved conjugate gradient scheme to the solution of least squares svm. *IEEE Transactions on Neural Networks*, 16(2):498–501, 2005.

[16] Kok Seng Chua. Efficient computations for large least square support vector machine classifiers. *Pattern Recognition Letters*, 24:75–80, 2003.

[17] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Large scale transductive svm. *Journal of Machine Learning Research*, 7:1687–1712, 2006.

[18] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[19] B. J. de Kruif and T. J. deVries. Pruning error minimization in least squares support vector machines. *IEEE Transactions on Neural Networks*, 14:696–702, 2003.

[20] Gregory Druck, Chris Pal, Andrew McCallum, and Xiaojin Zhu. Semi-supervised classification with hybrid generative/discriminative methods. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 280–289, New York, NY, USA, 2007. ACM.

[21] G. Fung and O. Mangasarian. Semi-supervised support vector machines for unlabeled data classification. *Optimization Methods and Software*, 15:29–44, 2001.

[22] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 529–536. MIT Press, Cambridge, MA, 2004.

[23] Licheng Jiao, Liefeng Bo, and Ling Wang. Fast sparse approximation for least square support vector machine. *IEEE Transactions on Neural Networks*, 18(3):685–697, 2007.

[24] Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

[25] Julia A. Lasserre, Christopher M. Bishop, and Thomas P. Minka. Principled hybrids of generative and discriminative models. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 87–94, Washington, DC, USA, 2006. IEEE Computer Society.

[26] J. Luts, J.A.K. Suykens, and S. Van Huffel. Semi-supervised learning: avoiding zero label assumptions in kernel based classifiers. Technical report, Internal Report 07-122, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2007.

[27] T. Mitchell. The role of unlabeled data in supervised learning. In *Proceedings of the Sixth International Colloquium on Cognitive Science, San Sebastian, Spain.*, 1999.

[28] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.

[29] E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.

[30] J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A.J. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, editors, *Advances in Large Margin Classiers*, pages 61–74. 2000.

[31] B. Scholkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[32] Matthias Seeger. Learning with labeled and unlabeled data. Technical report,

Technical Report , Institute for Adaptive and neural Computation, University of Edinburgh, February 2001.

[33] J. A. K. Suykens, L. Lukas, and L. Vandewalle. Sparse least squares support vector machine classifiers. In *European Symposium of Artificial Neural Networks*, 2000.

[34] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

[35] Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[36] I. Ulusoy and C. M. Bishop. Generative versus discriminative models for object recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition*, San Diego, 2005.

[37] T. Van Gestel, J. A. K. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and J. Vandewalle. Benchmarking least squares support vector machine classifiers. *Machine Learning*, 54(1):5–32, 2004.

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

[39] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.

[40] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2008. http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf.

ARTICLE 5

MODEL SELECTION FOR LS-SVM: APPLICATION TO HANDWRITING RECOGNITION

M. M. Adankon[1], and M. Cheriet[1],
[1]École de Technologie Supérieure, 1100 Notre-Dame Ouest,
Montréal, Québec, Canada H3C 1K3

Cet article est publié dans Pattern Recognition 42(12): 3264-3270.

Pattern Recognition ■■ (■■■■) ■■■–■■■

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Model selection for the LS-SVM. Application to handwriting recognition

Mathias M. Adankon*, Mohamed Cheriet

Synchromedia Laboratory for Multimedia Communication in Telepresence ETS, 1100 Notre-Dame-Ouest, Montréal, Canada H3C 1K3

ARTICLE INFO

Article history:
Received 7 August 2008
Accepted 15 October 2008

Keywords:
LS-SVM
Support vector machine
Model selection
Kernel machine

ABSTRACT

The support vector machine (SVM) is a powerful classifier which has been used successfully in many pattern recognition problems. It has also been shown to perform well in the handwriting recognition field. The least squares SVM (LS-SVM), like the SVM, is based on the margin-maximization principle performing structural risk minimization. However, it is easier to train than the SVM, as it requires only the solution to a convex linear problem, and not a quadratic problem as in the SVM. In this paper, we propose to conduct model selection for the LS-SVM using an empirical error criterion. Experiments on handwritten character recognition show the usefulness of this classifier and demonstrate that model selection improves the generalization performance of the LS-SVM.

1. Introduction

Support vector machines (SVMs) are particular classifiers which are based on the margin-maximization principle. They perform structural risk minimization, which was introduced to machine learning by Vapnik, and have yielded excellent generalization performance [1,2]. For nonlinear problems, SVMs use the kernel trick to produce nonlinear boundaries. The idea behind kernels is to map training data nonlinearly into a higher-dimensional feature space via a mapping function Φ and to construct a separating hyperplane which maximizes the margin. The construction of the linear decision surface in this feature space only requires the evaluation of dot products $\Phi(x)\cdot\Phi(y)=k(x,y)$, where $k(.)$ is called the kernel function [3–6].

The least-squares support vector machine (LS-SVM) is a variant of the standard SVM. It is the result of answering the following question: "How much can the SVM formulation be simplified without losing any of its advantages?" Suykens and Vandewalle [7] proposed an LS-SVM in which the training algorithm solves a convex problem like the SVM. It has been shown by a meticulous empirical study that the generalization performance of the LS-SVM is comparable to that of the SVM [8]. In addition, the training algorithm of the LS-SVM is highly simplified, since a linear problem is resolved instead of a quadratic programming (QP) problem in the SVM case.

Like the SVM, the LS-SVM is based on the margin-maximization principle that performs structural risk minimization; moreover, it inherits the SVMs generalization capacity. However, the choice of hyperparameters (regularization parameter and kernel parameters) affects the LS-SVMs performance. Thus, like the SVM and other classifiers, model selection is needed in order to obtain the best classifier performance. The classical method for choosing a good hyperparameter value is the cross-validation method based on the exhaustive search, but it becomes intractable when there are many hyperparameters (for example, if there are more than two kernel parameters). Several methods [2,9–17] have been developed for choosing the best hyperparameter values for the SVM classifiers. In 2001, Chapelle et al. [18] for the first time proposed an automatic method for selecting hyperparameters for SVMs using certain criteria which approximate the error of the leave-one-out (LOO) procedure. These criteria are called Span bound and radius-margin bound. Duan et al. have shown in Ref. [19] that the radius-margin bound gives a good prediction for the L2-SVM, but its practical viability is still not very satisfactory for the L1-SVM. Then, in 2003, Chung et al. [20] proposed modified radius-margin bound for L1-SVM. Recently, Ayat et al. [21] have proposed a new criterion based on empirical error, where an empirical estimate of the generalization error is minimized through a validation set. This criterion is a linear function which does not require the resolution of another quadratic problem except in SVM training.

In this paper, we propose to tune the LS-SVM hyperparameters using the empirical error criterion [21] in LOO cross-validation procedure, considering the useful work of Cawley [22]. Experimental results on a handwriting recognition problem demonstrate the usefulness of our method.

This paper is structured as follows. In Section 2, we provide a literature review of the LS-SVM. In Section 3, we present some of the strategies that have been proposed to improve the sparseness of the LS-SVM. In Section 4, we describe the LOO method as developed by

* Corresponding author.
E-mail addresses: mathias@livia.etsmtl.ca (M.M. Adankon), mohamed.cheriet@etsmtl.ca (M. Cheriet).

0031-3203/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2008.10.023

Cawley [22] and an automatic model selection for the LS-SVM based on empirical error minimization. In Section 5, we present our experimental results and discussion on a handwriting character recognition problems. In the last section, we conclude the paper.

2. Least squares support vector machines

We first consider a binary classification problem involving a dataset $\{(x_1,y_1),\ldots,(x_\ell,y_\ell)\}$ with $x_i \in \mathbb{R}^d$ and $y_i \in \{-1,1\}$. Nonlinear SVM classifiers use the kernel trick to produce nonlinear boundaries. The decision function given by an SVM is

$$f(x) = \text{sign}[w'\phi(x) + b] \tag{1}$$

where w and b are found by resolving the following optimization problem which expresses the maximization of the margin $2/\|w\|$ and the minimization of the training error:

$$\underset{w,b,\xi}{\text{minimize}} \quad \frac{1}{2}w'w + C\sum_{i=1}^{\ell}\xi_i \tag{2}$$

$$\text{subject to} \quad y_i[w'\phi(x)+b] \geq 1-\xi_i \quad \forall i = 1,\ldots,\ell \tag{3}$$

$$\xi_i \geq 0 \quad \forall i = 1,\ldots,\ell \tag{4}$$

The Lagrangian of the preceding problem is

$$\mathcal{L} = \frac{1}{2}w'w + C\sum_{i=1}^{\ell}\xi_i - \sum_{i=1}^{\ell}\alpha_i[y_i(w'\phi(x)+b) - 1 + \xi_i] - \sum_{i=1}^{\ell}\lambda_i\xi_i \tag{5}$$

with the Lagrange multipliers $\alpha_i \geq 0$ and $\lambda_i \geq 0$ for all $i = 1,\ldots,\ell$.

When we apply the Lagrange differentiation theorem, we obtain

$$f(x) = \text{sign}\left[\sum_{i=1}^{\ell}\alpha_i y_i k(x_i,x) + b\right] \tag{6}$$

with α solution to

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

$$\text{subject to} \quad \sum_{i=1}^{\ell}\alpha_i y_i = 0 \quad \text{and} \quad 0 \leq \alpha_i \leq C, \quad i = 1,\ldots,\ell \tag{7}$$

The LS-SVM is an interesting variant of the SVM proposed by Suykens et al. [7,23–25]. The standard SVM classifier of Vapnik is modified to transform the QP problem into a linear problem. These modifications are formulated as follows in the definition of the LS-SVM:

$$\underset{w,b,e}{\text{minimize}} \quad \frac{1}{2}w'w + \gamma\frac{1}{2}\sum_{i=1}^{\ell}e_i^2 \tag{8}$$

$$\text{subject to} \quad y_i[w'\phi(x_i)+b] = 1-e_i \quad \forall i = 1,\ldots,\ell \tag{9}$$

The original SVM formulation is modified at two points. First, the inequality constraints with the slack variable ξ_i expressed in Eq. (3) are replaced by the equality constraints with an error variable e_i. Second, a squared loss function is considered in the objective function. These two essential modifications simplify the problem, which becomes linear.

The Lagrangian of problem (8) is expressed by

$$\mathcal{L}(w,b,e,\alpha) = \frac{1}{2}w'w + \gamma\frac{1}{2}\sum_{i=1}^{\ell}e_i^2 - \sum_{i=1}^{\ell}\alpha_i[y_i[w'\phi(x)+b] - 1 + e_i] \tag{10}$$

where α_i are Lagrange multipliers.

The conditions for optimality yield

$$\begin{cases} \dfrac{\partial\mathcal{L}}{\partial w} = 0 \Rightarrow w = \displaystyle\sum_{j=1}^{\ell}\alpha_j y_j\phi(x_j) \\ \dfrac{\partial\mathcal{L}}{\partial b} = 0 \Rightarrow \displaystyle\sum_{j=1}^{\ell}\alpha_j y_j = 0 \\ \dfrac{\partial\mathcal{L}}{\partial e_j} = 0 \Rightarrow \alpha_j = \gamma e_j \quad \forall j = 1,\ldots,\ell \\ \dfrac{\partial\mathcal{L}}{\partial\alpha_j} = 0 \Rightarrow y_j[w'\phi(x_j)+b] - 1 + e_j = 0 \quad \forall j = 1,\ldots,\ell \end{cases}$$

We note that the system obtained from the Karush–Kuhn–Tucker conditions is linear. Its solution is found by solving the system of linear equations expressed in the following matrix form:

$$\begin{pmatrix} Q + \gamma^{-1}I & \bar{1}' \\ \bar{1} & 0 \end{pmatrix}\begin{pmatrix} \alpha \\ b \end{pmatrix} = \begin{pmatrix} Y \\ 0 \end{pmatrix} \tag{11}$$

where $Q_{ij} = k(x_i,x_j)$

$$Y = (y_1,\ldots,y_\ell)'$$

$$\alpha = (\alpha_1,\ldots,\alpha_\ell)'$$

$$\bar{1} = (1,\ldots,1)$$

For training the LS-SVM, Suykens et al. [7,23] proposed an algorithm based on the conjugate gradient technique. In Ref. [26], an improved algorithm, also based on the conjugate gradient, is developed with a reduced computational time. Among the methods for training the LS-SVM, are the SMO technique adapted to find the LS-SVM solution and an algebraic method proposed by Chua in Ref. [27].

3. Sparse LS-SVM

The main problem associated with the LS-SVM is its lack of sparseness. Unlike the SVM, almost all the training points are support vectors, $\alpha \neq 0$. Moreover, the kernel expansions of the LS-SVM are highly dense. This makes it difficult to use the LS-SVM in large-scale applications.

To overcome this limitation of the LS-SVM, we could suggest training the LS-SVM in the primal space by using the kernel PCA strategy for reducing the dimensionality. In the literature, in order to reduce the complexity of the original LS-SVM, several methods have been proposed for reducing existing support vector expansions or training the classifier with a reduced set [28–32].

Suykens et al. [28] proposed to prune training examples that have smaller absolute support value. In Ref. [30], a more sophisticated pruning method was proposed, in which the examples that introduced the smallest approximate error are removed from the training set. However, these two pruning methods keep outliers, which has a negative influence on the classifier in the reduced training set. Also, the former method removes the samples near the boundary from the training set, and this decreases the performance of the classifier. Li et al. [31] overcome these two problems by proposing certain heuristics.

In Refs. [29,33], a different way to obtain a sparse LS-SVM is proposed. The authors suggest first selecting reference vectors to form a basis for the subspace populated by the data. They then modify the objective function by limiting the kernel expansions to the preselected subspace. The main difference between this and other methods is the fact that the whole training set is used, although a set of candidate support vectors is preselected. The objective function

Please cite this article as: M.M. Adankon, M. Cheriet, Model selection for the LS-SVM. Application to handwriting recognition, Pattern Recognition (2008), doi: 10.1016/j.patcog.2008.10.023

proposed in Refs. [29,33] is as follows:

$$\mathcal{L}(\alpha,b)=\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j k(z_i,z_j)+\gamma\sum_{i=1}^{\ell}\left(y_i-\sum_{j=1}^{n}\alpha_j k(x_i,z_j)-b\right)^2 \tag{12}$$

where $n \ll \ell$ is the number of reference vectors selected z_i.

This optimization problem like the original LS-SVM can be expressed in the form of a system of linear equations as

$$(R+Z'Z)\begin{pmatrix}\alpha\\ b\end{pmatrix}=Z'Y \tag{13}$$

with $Z=[\hat{K},\vec{1}]$, and where $\hat{K}$ is a $\ell \times n$ matrix, the element of which is $\hat{k}(x_i,z_j)$ and

$$R=\begin{pmatrix}\gamma^{-1}K & \vec{0}'\\ \vec{0} & 0\end{pmatrix}$$

where K is a $n \times n$ matrix,the element of which is $k(z_i,z_j)$.

There are many approaches to select a set of reference vectors [5,29,34], but almost all of them have a heavy computational cost.

Recently, Jiao et al. [32] proposed a fast sparse approximation for LS-SVM (FSALS-SVM), in which the classifier is iteratively built by adding one basis function from a kernel-based dictionary. The basis function is selected in order to obtain the largest decrease in the LS-SVM objective function. The process is ended by using a flexible and stable criterion. The algorithm proposed in Ref. [32] is fast and exhibits good generalization performance.

4. Model selection for the LS-SVM

Model selection for the LS-SVM consists of selecting the hyperparameters that yield the best performance of the machine. The LS-SVM classifier has two types of hyperparameters: the regularization parameter γ, which controls the tradeoff between training error minimization and margin maximization, and the kernel parameters that define the given kernel function.

4.1. Exact LOO

LOO is the special case of cross-validation. In k-fold cross-validation, we divide the available training data into k subsets. We train the machine k times, each time leaving out one of the subsets from training, and use only the omitted subset to compute the given error criterion. If k equals the training set size, this is called "LOO" cross-validation or LOO cross-validation.

Let $[\alpha^{(-i)};b^{(-i)}]$ represent the parameters of the LS-SVM when the ith sample is omitted during the LOO cross-validation procedure. It is shown that

$$\begin{pmatrix}\alpha^{(-i)}\\ b^{(-i)}\end{pmatrix}=H_{(-i)}^{-1}[y_1,\ldots,y_{i-1},y_{i+1},\ldots,y_\ell,0]' \tag{14}$$

where

$$H=\begin{pmatrix}Q+\gamma^{-1}I & \vec{1}'\\ \vec{1} & 0\end{pmatrix} \tag{15}$$

and $H_{(-i)}$ is the matrix obtained when the ith sample is omitted in H.

After some manipulations and using the block matrix inversion lemma (see Ref. [22] for details), the following result is obtained:

$$y_i-f^{(-i)}(x_i)=\frac{\alpha_i}{H_{ii}^{-1}} \tag{16}$$

where $f^{(-i)}(x_i)$ is the LOO prediction for the ith sample.

1. *Initialize the learning rate*
2. *Initialize the hyperparameters*
3. *Repeat until convergence*
   - 3.1 *Train the LS-SVM*
   - 3.2 *Compute the gradient of error*
   - 3.3 *Estimate the learning rate*
   - 3.4 *Update the hyperparameters*

Fig. 1. Descent gradient algorithm for the LS-SVM hyperparameters optimization.

So, without computing ℓ times the parameter of the machines, it is possible to compute the criterion error as the predicted residual sum-of-squares (*PRESS*):

$$PRESS=\sum_{i=1}^{\ell}[y_i-f^{(-i)}(x_i)]^2 \tag{17}$$

Similar work on the sparse LS-SVM is performed in Ref. [33]. So, LOO cross-validation becomes a practical strategy for model selection for the LS-SVM while it is intractable with the SVM.

4.2. Empirical error minimization

In this section, we describe model selection for the LS-SVM using the empirical error criterion developed in Refs. [17,21] for tuning the kernel parameters of the original SVM classifier.

Let us define $t_i=(y_i+1)/2$. The empirical error is given by the following expression:

$$E_i=|t_i-\tilde{p}_i| \tag{18}$$

where $\tilde{p}_i$ is the estimated posterior probability corresponding to the data example x_i.

The estimated posterior probability is determined by

$$\tilde{p}_i=\frac{1}{1+\exp(A\cdot f_i+B)} \tag{19}$$

where $f_i=f(x_i)$ and parameters A and B are fitted after minimizing the cross-entropy error [35] as proposed by Platt in Ref. [36]. In this paper, we use fixed values for A and B, because we need to keep the continuity of the empirical error expression for each validation sample at each iteration.

We assume that the kernel function depends on one or several parameters, encoded within the vector $\theta=(\theta_1,\ldots,\theta_n)$ including the hyperparameter γ. The optimization of these parameters is performed by a gradient descent minimization algorithm [37] where the objective function is $E=\sum E_i$ (see Fig. 1). However, sometimes our problem is not convex, and, to overcome this situation, we use many starting points. We can also use the simple function *fminsearch* implemented in Matlab with different starting points.

Despite the fact that the empirical error criterion was first developed for the SVM, its use with the LS-SVM does not simply involve a direct mapping. So, the derivative computing for the LS-SVM needs a careful analysis of its model.

The derivative of the empirical error with respect to θ is evaluated using the validation dataset. Let us assume N to be the size of the

Please cite this article as: M.M. Adankon, M. Cheriet, Model selection for the LS-SVM. Application to handwriting recognition, Pattern Recognition (2009), doi:10.1016/j.patcog.2008.10.023

validation dataset. Then

$$\frac{\partial E}{\partial \theta} = \frac{\partial}{\partial \theta}\left(\frac{1}{N}\sum_{i=1}^{N} E_i\right) = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial E_i}{\partial \theta} \tag{20}$$

E_i is expressed in terms of the estimated posterior probability, which depends on the output f_i of the LS-SVM. Therefore, we can write

$$\frac{\partial E_i}{\partial \theta} = \frac{\partial E_i}{\partial \hat{p}_i} * \frac{\partial \hat{p}_i}{\partial f_i} * \frac{\partial f_i}{\partial \theta} \tag{21}$$

According to Eq. (18), we have

$$\frac{\partial E_i}{\partial \hat{p}_i} = \frac{\partial |t_i - \hat{p}_i|}{\partial \hat{p}_i} = -y_i \tag{22}$$

Considering Eq. (19), which gives the expression of the estimated posterior probability, the second part of the gradient is

$$\frac{\partial \hat{p}_i}{\partial f_i} = -A\hat{p}_i(1-\hat{p}_i) \tag{23}$$

For evaluating the last part of the gradient, we consider the expression of the LS-SVM output as

$$\frac{\partial f_i}{\partial \theta} = \frac{\partial}{\partial \theta}\left(\sum_{j=1}^{\ell} a_j y_j k(x_j, x_i) + b\right) = \sum_{j=1}^{\ell} y_j \left[\frac{\partial k(x_j, x_i)}{\partial \theta} x_j + \frac{\partial x_j}{\partial \theta} k(x_j, x_i)\right] + \frac{\partial b}{\partial \theta} \tag{24}$$

In Eq. (24), the evaluation of the derivative $\partial k(x_j, x_i)/\partial\theta$ is easy and depends on the type of the kernel to be chosen. But, for estimating the derivative $\partial x_j/\partial\theta$, we need the expression of the terms x_j with respect to θ. For this, we use Eq. (11):

$$\tilde{a} = H^{-1}\tilde{Y} \quad \text{where} \quad \tilde{x} = (x; b) \quad \text{and} \quad \tilde{Y} = (Y; 0)$$

Then

$$\frac{\partial \tilde{x}}{\partial \theta} = \frac{\partial H^{-1}}{\partial \theta}\tilde{Y} + H^{-1}\frac{\partial \tilde{Y}}{\partial \theta} = \frac{\partial H^{-1}}{\partial \theta}\tilde{Y}$$

For computing the components of $\partial H^{-1}/\partial\theta$, we use the matrix relation proposed in Ref. [37]:

$$\frac{\partial \tilde{x}}{\partial \theta} = -H^{-1}\frac{\partial H}{\partial \theta}H^{-1}\tilde{Y} = -H^{-1}\frac{\partial H}{\partial \theta}\tilde{a} \tag{25}$$

It is also possible to minimize the empirical error through the LOO cross-validation procedure. Considering Eq. (16), the LOO prediction for the ith sample is expressed by

$$f^{(-i)}(x_i) = y_i - \frac{a_i}{H^{-1}_{ii}} \tag{26}$$

Then, using the previous equality, we can compute the empirical error for each training sample by using the LOO cross-validation procedure.

4.3. Complexity analysis

The computational complexity of the basic training algorithm is dominated by evaluating the inverse of the LS-SVM matrix H, which is $\mathcal{O}(\ell^3)$ operations. Evaluation of the inverse of matrix H is very useful for computing the LOO procedure and the empirical error gradient. Thus, the additional complexity time needed to compute the gradient is $\mathcal{O}(\ell^2)$, which is less than the training time. Note that, if the sparse LS-SVM is used, the total complexity is reduced and equal to $\mathcal{O}(n^2\ell + n^2) \approx \mathcal{O}(n^2\ell)$, where n represents the number of support vectors that is usually smaller than the training size.

5. Experiments

We tested our model selection algorithm for the LS-SVM on a handwriting recognition problem with the USPS and MNIST databases.

5.1. USPS database

USPS is the well-known US Postal Service handwritten digits recognition corpus. The digits are represented by normalized gray scale images of size 16×16. The learning dataset contains 7291 samples (5291 for training and 2000 for validation), while the testing dataset consists of 2007 other samples.

5.1.1. Experimental setup

We have a multi-class problem with 10 classes. Then, we train 10 machines using the *one-against-all* strategy. We test the original LS-SVM and the sparse LS-SVM proposed in Ref. [32], the code for which is available at http://see.xidian.edu.cn/graduate/jlfbo. We use the RBF kernel. For each classifier, the hyperparameters are optimized by using the *PRESS* criterion with the LOO strategy and empirical error criterion. Finally, we compute the mean of the hyperparameters with 10 values obtained from each of the 10 classifiers.

5.1.2. Results and discussion

The results obtained are presented in Table 1, where we report the results obtained with various model selection techniques. Our results, obtained by empirical error minimization, are similar to those obtained by the classical grid search method, although the latter is quite costly in terms of computing time and becomes intractable when there are more than two hyperparameters. Note that the *PRESS* statistic that is best suited to regression problems [22,33], is less adequate for the LS-SVM classifier.

Also, we note that the sparse LS-SVM achieved good generalization in comparison with the original LS-SVM. Moreover, the classifier obtained with the sparse LS-SVM algorithm has fewer support vectors, which reduces the complexity of the machine. Thus, the regularization of the classifier is reinforced. In Fig. 2, we plot the test error rate and the number of support vectors according to the model selection method with the sparse LS-SVM. We point out that the LOO procedure with the empirical error criterion give the best result in terms of the machine complexity and generalization, thereby confirming our approach.

In Table 2, we compare the sparse LS-SVM with the SVM, where we used the empirical error criterion for model selection. This experiment shows that the sparse LS-SVM is a good alternative to the SVM. The results point up the good performance of the sparse LS-SVM, in terms of accuracy, complexity, and sparseness. Moreover, the classifier yields the best generalization performance on this problem. Note that the performance of the LS-SVM classifier comes from the good model selection achieved by the LOO procedure with empirical error criterion, which it is not possible to do with the original SVM classifier.

Table 1
The error rate in percent obtained on the test set with the original LS-SVM and the sparse LS-SVM.

	Original LS-SVM	Sparse LS-SVM
Empirical error criterion	4.63	4.18
LOO with PRESS criterion	4.48	4.43
LOO with empirical error	4.53	4.03
Grid search method	4.43	4.34

We present the results with four different model selection techniques: empirical error criterion with a validation set, LOO with the PRESS criterion [22], LOO with the empirical error criterion, and the classical grid search method done in Ref. [32].

Please cite this article as: M.M. Adankon, M. Cheriet, Model selection for the LS-SVM. Application to handwriting re..., cognition, Pattern Recognition (2008), doi: 10.1016/j.patcog.2008.10.023

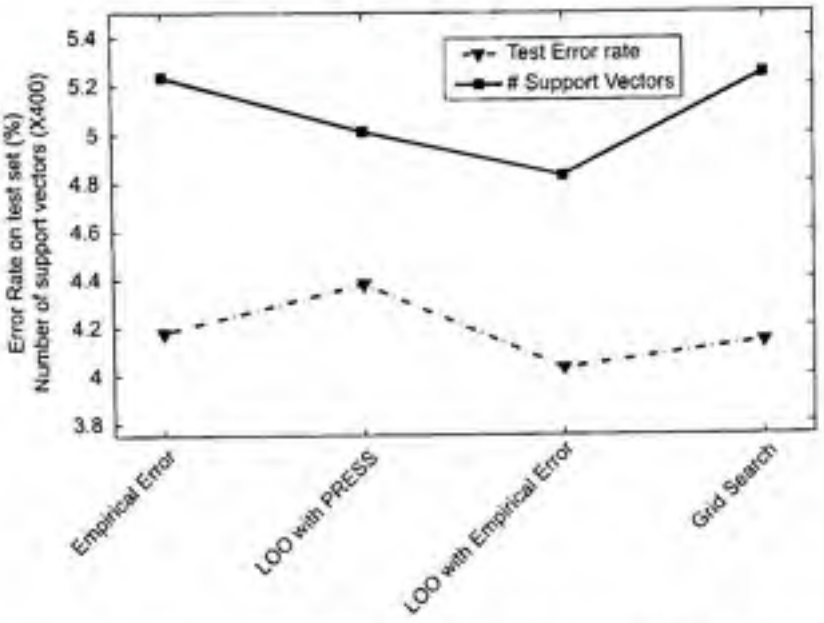


Fig. 2. The test error rate and the number of support vectors according to the model selection method with the sparse LS-SVM.

Table 2
Comparison between the SVM and the sparse LS-SVM: we train the SVM with Joachims' algorithm, which is called SVMlight.

	Original SVM	Sparse LS-SVM
Training time (s)	45	44
Number of support vectors	5069	1932
Testing time (s)	5.32	2.21
Test error (%)	4.33	4.03

5.2. MNIST database

In this section, we tested our algorithms on a second isolated handwriting recognition problem, using the MNIST database. The MNIST (modified NIST) database [38] is a subset extracted from the NIST database. The digits have been size-normalized and centered on a fixed-size image. The learning dataset contains 60,000 samples (50,000 for training and 10,000 for validation) while the testing dataset consists of 10,000 other samples.

5.2.1. Experimental setup

We use a pairwise coupling strategy for this problem, also known as a *one-against-one* strategy. We then train 45 classifiers, and each SVM is optimized by combining the LOO procedure with the empirical error criterion. We used the RBF kernel.

For the test phase, we applied different types of couplings after mapping the 45 SVM outputs into probabilities. The probability that a given observation belongs to class ω_i $(i = 1, \ldots, 10)$ is

$$p_i = \frac{1}{45} \sum_{i \neq j} \sigma(p_{ij}) \tag{27}$$

where

$$p_{ij} = P(x \in \omega_i / x \in \omega_i \cup \omega_j)$$

and σ is a coupling function. For a complete description see Ref. [39] where the following functions are reported:

$$\text{PWC1} \quad \sigma(x) = \begin{cases} 1 & \text{if } x > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Table 3
The error rate in percent obtained on the MNIST dataset using several types of couplings with the original SVM and the sparse LS-SVM.

Coupling model	Original SVM	Sparse LS-SVM
PWC1	1.6	1.40
PWC2	1.5	1.33
PWC3	1.7	1.37
PWC4	1.6	1.37
PWC5	1.5	1.40

The results with the original SVM are taken from Ref. [17] where model selection is performed using the empirical error criterion.

$$\text{PWC2} \quad \sigma(x) = x$$

$$\text{PWC3} \quad \sigma(x) = \frac{1}{1 + e^{-12(x-0.5)}}$$

$$\text{PWC4} \quad \sigma(x) = \begin{cases} 1 & \text{if } x > 0.5 \\ x & \text{otherwise} \end{cases}$$

$$\text{PWC5} \quad \sigma(x) = \begin{cases} x & \text{if } x > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

5.2.2. Results and discussion

We test the sparse LS-SVM with the LOO procedure combined with the empirical error. The results are reported in Table 3, along with those obtained with the original SVM in Ref. [17]. These results reveal once again that the generalization performance of the sparse LS-SVM is competitive with that of original SVM. They confirm the usefulness of LS-SVM model selection for building a classifier for the handwriting recognition problem. In summary, the empirical error is a good model selection criterion which, combined with the LOO procedure, provides a strong basis for computing model selection for the LS-SVM.

5.3. Comparison with related works

In the handwriting recognition problem, to our knowledge, no work on LS-SVM model selection has been carried out to date to

Please cite this article as: M.M. Adankon, M. Cheriet, Model selection for the LS-SVM. Application to handwriting re..., cognition Pattern Recognition (2008), doi: 10.1016/j.patcog.2008.10.023

Table 4
Details of the Splice, Image, German, and Flare solar datasets taken from the UCI benchmark repository

Dataset name	No. of features	No. of training patterns	No. of testing patterns	No. of realizations
Splice	60	1000	2175	20
Image	18	1300	1010	20
German	20	700	300	100
Flare solar	9	666	400	100

Table 5
Error rates of the LS-SVM classifier with different model selection techniques

Dataset name	LS-SVM	LS-SVM-BR	Sparse LS-SVM-EE
Splice	10.97 ± 0.158	10.91 ± 0.154	11.31 ± 0.131
Image	3.00 ± 0.158	2.90 ± 0.154	2.78 ± 0.149
German	23.55 ± 0.216	23.59 ± 0.216	23.73 ± 0.216
Flare solar	34.22 ± 0.169	34.07 ± 0.171	33.31 ± 0.153

The LS-SVM and the LS-SVM-BR techniques are reported in Ref. [40]; LS-SVM-EE is our method where the sparse LS-SVM classifier hyperparameters are optimized with the LOO strategy using the empirical error criterion.

compare with our method. However, and for the sake of comparison, we will compare it with the recent work of Cawley and Talbot [40] on the UCI benchmark datasets. We used the datasets with a larger training size than 500: Splice, Image, German, and Flare solar. Each dataset describes a two-class problem and is randomly partitioned many times in order to constitute the training and the testing sets.

For the Splice and Image datasets, the partitioning was performed 20 times, but in the case of the German and Flare solar datasets, it was performed 100 times. See Table 4 for further details on the datasets. We perform model selection independently for each realization, following the same setup as in Ref. [40] with the RBF kernel. The results are reported in Table 5, where we report the results obtained with the various model selection techniques developed in Ref. [40]. In this last paper, it is proposed to use Bayesian regularization at the second level of inference, adding a regularization term to the model selection criterion. This strategy is applied to the LOO cross-validation procedure with the LS-SVM classifier.

Table 5 presents the error rates of the LS-SVM classifier with the various model selection techniques. We reported the mean and the standard error computed over the 20 or 100 realizations for each dataset. We note that our results are similar to those obtained in Ref. [40]. However, in terms of sparseness, our model is more robust and more effective for large datasets.

6. Conclusion

The LS-SVM classifier, like other kernel machines, gives a poor generalization when the hyperparameters are not tuned efficiently. In this paper, we propose a model selection strategy for the LS-SVM which is a variant of the popular SVM classifier. We performed model selection using the empirical error criterion through the LOO procedure. We applied our algorithms on a handwriting recognition problem, which gave promising results. Compared with the SVM, the sparse LS-SVM classifier, empowered by model selection based on the empirical error criterion and the LOO procedure, achieved higher performance. We conclude from this that the sparse LS-SVM with model selection would be an interesting alternative classifier for the SVM in pattern recognition systems.

Acknowledgments

The authors would like to thank the anonymous reviewers of the ICFHR for their helpful comments and NSERC of Canada for its financial support and for providing a Research Scholarship.

References

[1] V.N. Vapnik, Principles of risk minimization for learning theory, Advances in Neural Information Processing Systems, vol. 4, Morgan Kaufman, San Mateo, CA, 1992, pp. 831–838.
[2] V.N. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[3] B.E. Boser, I. Guyon, V. Vapnik, A training algorithm for optimal margin classifiers, Comput. Learn. Theory (1992) 144–152.
[4] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, Cambridge, 2000.
[5] B. Scholkopf, A.J. Smola, Learning with Kernels, MIT Press, Cambridge, MA, 2002.
[6] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, Cambridge, 2004.
[7] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, Least Squares Support Vector Machines, World Scientific, Singapore, 2002.
[8] T. Van Gestel, J.A.K. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, J. Vandewalle, Benchmarking least squares support vector machine classifiers, Machine Learning 54 (1) (2004) 5–32.
[9] G. Wahba, Y. Lin, H. Zhang, Generalized approximate cross validation for support vector machines, or, another way to look at margin-like quantities, Technical report, Department of Statistics, University of Wisconsin, February 25 1999.
[10] T.S. Jaakkola, D. Haussler, Probabilistic kernel regression models, in: Workshop in Conference on Artificial Intelligence and Statistics, 1999.
[11] T. Joachims, Estimating the generalization performance of a SVM efficiently, in: International Conference on Machine Learning, 2000, pp. 431–438.
[12] M. Opper, O. Winther, Mean field methods for classification with Gaussian processes, in: The 1998 Conference on Advances in Neural Information Processing Systems II, MIT Press, Cambridge, MA, 1999, pp. 309–315.
[13] M. Opper, O. Winther, Gaussian processes and SVM: mean field and leave-one-out, in: A.J. Smola, P.L. Bartlett, B. Scholkopf, D. Schuurmans (Eds.), Advances in Large Margin Classifiers, MIT Press, Cambridge, MA, 2000, pp. 311–326.
[14] O. Chapelle, V. Vapnik, Model selection for support vector machines, in: Advances in Neural Information Processing Systems, 1999.
[15] V. Vapnik, O. Chapelle, Bounds on error expectation for support vector machines, Neural Comput. 12 (9) (2000).
[16] C. Gold, P. Sollich, Fast Bayesian support vector machine parameter tuning with the nystrom method, in: IJCNN05, 2005, pp. 2820–2825.
[17] M.M. Adankon, M. Cheriet, Optimizing resources in model selection for support vector machines, Pattern Recognition 40 (3) (2007) 953–963.
[18] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Machine Learning 46 (1) (2002) 131–159.
[19] K. Duan, S. Keerthi, A.N. Poo, Evaluation of simple performance measures for tuning SVM hyperparameters, Neurocomputing 51 (2003) 41–59.
[20] K.-M. Chung, W.-C. Kao, L.-L. Wang, C.-L. Sun, C.-J. Lin, Radius margin bounds for support vector machines with the RBF kernel, Neural Comput. 15 (2003) 2643–2681.
[21] I.N.E. Ayat, M. Cheriet, C.Y. Suen, Automatic model selection for the optimization of the SVM kernels, Pattern Recognition 38 (10) (2005) 1733–1745.
[22] G. Cawley, Leave-one-out cross-validation based model selection criteria for weighted LS-SVMs, in: Proceedings IJCNN 2006, Vancouver, Canada, July 2006.
[23] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (3) (1999) 293–300.
[24] G. Lanckriet, A. Lambrechts, B. De Moor, J. Vandewalle, T. Van Gestel, J. Suykens, Bayesian framework for least squares support vector machine classifiers, Gaussian processes and kernel fisher discriminant analysis, Neural Comput. 15 (5) (2002) 1115–1148.

Please cite this article as: M.M. Adankon, M. Cheriet, Model selection for the LS-SVM. Application to handwriting recognition, Pattern Recognition (2008), doi: 10.1016/j.patcog.2008.10.023

[25] S.-F. Zheng, Least square support vector machine and its Bayesian interpretation, Technical report, June 2004.
[26] C.J. Ong, W. Chu, S.S. Keerthi, An improved conjugate gradient scheme to the solution of least squares SVM, IEEE Trans. Neural Networks 16 (2) (2005) 498–501.
[27] K.S. Chua, Efficient computations for large least square support vector machine classifiers, Pattern Recognition Lett. 24 (2003) 75–80.
[28] L. Lukas, J.A.K. Suykens, J. Vandewalle, Sparse least squares support vector machine classifiers, in: European Symposium of Artificial Neural Networks, 2000.
[29] G.C. Cawley, N.L.C. Talbot, Improved sparse least-squares support vector machines, Neurocomputing 48 (2002) 1025–1031.
[30] B.J. de Kruif, T.J. deVries, Pruning error minimization in least squares support vector machines, IEEE Trans. Neural Networks 14 (2003) 696–702.
[31] C. Lin, Y. Li, W. Zhang, Improved sparse least-squares support vector machine classifiers, Neurocomputing 69 (2006) 1655–1658.
[32] L. Jiao, L. Bo, L. Wang, Fast sparse approximation for least square support vector machine, IEEE Trans. Neural Networks 18 (3) (2007) 685–697.
[33] G.C. Cawley, N.L.C. Talbot, Fast exact leave-one-out cross-validation of sparse least-squares support vector machines, Neural Networks 17 (2004) 1467–1475.
[34] G. Baudat, F. Anouar, Feature vector selection and projection using kernels, Neurocomputing 55 (2003) 31–38.
[35] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, Great Britain, 1995.
[36] J. Platt, Probabilistic outputs for support vector machines and comparison to regularized likelihood methods, in: A.J. Smola, P. Bartlett, B. Schoelkopf, D. Schuurmans (Eds.), Advances in Large Margin Classifiers, 2000, pp. 61–74.
[37] Y. Bengio, Gradient-based optimization of hyper-parameters, Neural Comput. 12 (8) (2000) 1889–1900.
[38] Y. LeCun, L. Bottom, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (1998) 2278–2324.
[39] M. Moreira, E. Mayoraz, Improved pairwise coupling classification with correcting classifiers, in: ECML, 1998, pp. 160–171.
[40] G. Cawley, N. Talbot, Preventing over-fitting during model selection via Bayesian regularisation of the hyper-parameters, J. Mach. Learn. Res. 8 (2007) 841–861.

About the Author—MATHIAS M. ADANKON received the B.Ing. degree in Electrical Engineering from the University of Abomey-Calavi, Benin. He received the Master's degree in Automated Manufacturing Engineering from the Ecole de Technologie Superieure, the University of Quebec, Montreal, Canada. He is currently a Ph.D. student at LIVIA (Laboratory for Imagery, Vision and Artificial Intelligence) and at Synchromedia Laboratory for Multimedia Communication in Telepresence, University of Quebec. His research interests are centered on Classification using kernel method, pattern recognition, and machine learning.

About the Author—MOHAMED CHERIET was born in Algiers, Algeria, in 1960. He received his B.Eng. from USTHB University, Algiers in 1984 and his M.Sc. and Ph.D. degrees in Computer Science from the University of Pierre et Marie Curie, Paris VI in 1985 and 1988, respectively. Since 1992, he has been a professor in the Automation Engineering Department at the École de Technologie Supérieure, University of Quebec, Montreal, and was appointed as a full professor there in 1998. He co-founded the Laboratory for Imagery, Vision and Artificial Intelligence (LIVIA) at the University of Quebec, and was its director from 2000 to 2006. He also founded the SYNCHROMEDIA Consortium (Multimedia Communication in Telepresence) there, and has been its director since 1998. His interests include document image analysis, OCR, mathematical models for image processing, pattern classification models and learning algorithms, as well as perception in computer vision. Dr. Cheriet has published more than 200 technical papers in the field, and has served as chair or co-chair of the following international conferences: VI'1998, VI'2000, IWFHR'2002, and ICFHR'2008. He currently serves on the editorial board and is an associate editor of several international journals: IJPRAI, IJDAR, and Pattern Recognition. He co-authored a book entitled, "Character Recognition Systems: A guide for Students and Practitioners," John Wiley and Sons, Spring 2007. Dr. Cheriet is a senior member of the IEEE and the chapter chair of IEEE Montreal Computational Intelligent Systems (CIS).

Please cite this article as: M.M. Adankon, M. Cheriet, Model selection for the LS-SVM. Application to handwriting re..., cognition Pattern Recognition (2008), doi: 10.1016/j.patcog.2008.10.023