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
NEDJEM EDDINE AYAT

SÉLECTION AUTOMATIQUE DE MODÈLE DANS LES MACHINES À VECTEURS DE SUPPORT: APPLICATION À LA RECONNAISSANCE D'IMAGES DE CHIFFRES MANUSCRITS

MONTRÉAL, LE 13 JANVIER 2004

CETTE THÈSE A ÉTÉ ÉVALUÉE

PAR UN JURY COMPOSÉ DE :

M. Mohamed Cheriet, directeur de recherche
Département de génie de la production automatisée à l'École de Technologie Supérieure.

M. Ching Y. Suen, codirecteur
Centre for Pattern Recognition and Machine Intelligence, Concordia University.

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

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

M. Amar Mitiche, examinateur
Institut National de Recherche Scientifique, Université du Québec.

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

LE 10 DÉCEMBRE 2003

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

SÉLECTION AUTOMATIQUE DE MODÈLE DANS LES MACHINES À VECTEURS DE SUPPORT: APPLICATION À LA RECONNAISSANCE D'IMAGES DE CHIFFRES MANUSCRITS

Nedjem Eddine Ayat

SOMMAIRE

La problématique étudiée dans cette thèse est la sélection de modèle automatique des machines à vecteurs de support (SVM) pour la reconnaissance de chiffres manuscrits. Nous présentons une nouvelle méthodologie de sélection de modèle automatique du SVM sur des données binaires et multiclasse. L'approche permet d'optimiser les paramètres de noyaux et de réduire efficacement la complexité du classifieur en minimisant le nombre de vecteurs de support. Ceci s'accompagne d'une réduction drastique de l'erreur de généralisation.

La méthodologie proposée est basée sur un nouveau critère de sélection de modèle estimant la probabilité d'erreur du SVM. Ce critère est ensuite minimisé en utilisant une procédure efficiente de descente de gradient. La probabilité d'erreur est une erreur empirique estimée sur des observations de validation représentant le même problème de classification. Son calcul utilise les estimations des probabilités à posteriori de ces observations.

Pour des fins de comparaison, nous considérons deux autres critères de sélection de modèle que sont le GACV et le VC. Ce sont deux critères analytiques approximant des bornes supérieures de l'erreur. Pour le premier, nous proposons aussi un nouvel algorithme de minimisation. Les expériences effectuées sur un problème de classification binaire montrent la supériorité du critère de l'erreur empirique et sa capacité à sélectionner adéquatement les hyper-paramètres du SVM. Aussi, le critère garantit la solution de moindre complexité en produisant le plus faible nombre de vecteurs de support.

Par ailleurs, sur des données multiclasses, nous proposons deux approches de sélection automatique de modèle dans la stratégie un-contre-un. La première, dite *«approche locale»* permet d'optimiser l'ensemble de SVM individuellement en adaptant leurs hyper-paramètres aux données du couple de classes considéré. La deuxième, dite *«approche globale»*, permet d'optimiser simultanément l'ensemble de SVM en prenant en compte le comportement de chacun d'eux. Pour cette dernière, nous proposons de maximiser la vraisemblance des données de validation à travers l'ensemble de SVM en minimisant l'erreur quadratique entre les probabilités à posteriori des classes et les probabilités désirées des données. Les deux approches sont validées expérimentalement sur des données réelles issues des bases d'images de chiffres manuscrits arabes USPS et indiens INDCENPARMI.

AUTOMATIC MODEL SELECTION FOR SUPPORT VECTORS MACHINES : APPLICATION TO THE RECOGNITION OF IMAGES OF HANDWRITTEN DIGITS

Nedjem Eddine Ayat

ABSTRACT

This thesis relates to the automatic model selection of Support Vectors Machine (SVM) for the recognition of handwritten digits. We propose a new model selection methodology for the automatic optimization of SVM classifier on binary and multiclass data. The approach allows to optimize the kernel parameters and to reduce efficiently the number of support vectors. This goes along with a drastic reduction of the generalization error.

The proposed methodology suggests to use a new model selection criterion based on the estimation of the probability of error of the SVM classifier. Thereby, the criterion is minimized using an efficient gradient descent framework. The probability of error is an empirical error estimate computed over an independent validation set through the approximation of the posterior probabilities of the data examples.

For the sake of comparison, we consider two other model selection criterions which are GACV and VC. These are algebraic criterions which approximate upper bounds of the expected error. For the former, we also propose a new minimization scheme. The experiments done on a binary classification problem prove that the empirical error criterion is able to choose adequately the SVM hyper-parameters. Moreover, the criterion guarantees a less complex model with the lowest number of support vectors.

We also undertook the problem of SVM kernels optimization when multiple data class labels are available. So, we propose two model selection strategies adapted for the one-against-one data partitioning. A *'local approach'* optimizes individually the provided set of pairwise classifiers by adapting their hyper-parameters to the related classes. In the other side, a *'global approach'*, allows to optimize simultaneously the whole set of SVM classifiers by taking into account the behavior of each of them. For the latter, we propose to maximize the likelihood of the validation data by minimizing the quadratic error between the estimated classes' posterior probabilities and the desired ones. The approaches are then tested and evaluated on synthetic and real life data. The experiments done on real data include a whole set of experiments on images of handwritten digits from USPS database and of Indian digits from INDCENPARMI database.

REMERCIEMENTS

Mes vifs remerciements vont à mon directeur de thèse, le Professeur Mohamed Cheriet pour son suivi, pour son soutien et pour ses vifs encouragements, ainsi que pour les conseils avisés qu'il m'a prodigué tout au long de cette thèse.

Je remercie également mon codirecteur de thèse, le Professeur Ching Y. Suen pour son suivi et pour l'aide précieuse qu'il m'a fourni au sein du CENPARMI.

Aussi, je tiens à remercier Alain Biem, Chercheur Senior chez IBM à Watson (USA) et Amar Mitiche, Professeur à l'INRS, pour s'être intéressés à mon travail et avoir accepté de faire partie de mon jury et examiner mon travail. Enfin, ma gratitude au Professeur Richard Lepage qui a accepté de présider ce jury et organiser ma soutenance.

Tous mes remerciements vont également à tous les membres passés et présents du LIVIA et du CENPARMI avec qui j'ai passé des moments agréables, plus particulièrement à Lakhdar Remaki, Jonathan, Ali, Youssef, Frédéric, Luis, Luiz, Marisa, Flavio, Alexandro, Guillaume, Yousef Al-Ohali et Karim Abou-Moustafa. Un grand merci va également à Horia qui m'a aidé pendant les corrections du manuscrit, à Omar Msaaf pour m'avoir prodigué ses précieux conseils quotidiens, sans oublier Fatiha Anouar et Gaston Baudat de Mars Electronics.

Enfin, je dédie ce travail à toute ma famille en Algérie et en France, à Luna ainsi qu'à Jalil et Hocine.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS ET SIGLES

SVM	Machine à Vecteurs de Support.
RBF	Noyau à base radiale.
KMOD	«Kernel with Moderate Decreasing».
PMC	Perceptron multicouche.
VC	Dimension de Vapnik-Chernovenkis.
GACV	Erreur de validation croisée généralisée.
L1	Norme L1.
L2	Norme L2.
USPS	«United States Postal Service».
CENPARMI	«Centre For Pattern Recognition and Machine Intelligence».
INDCENPARMI	Base d'images de chiffres Indiens de CENPARMI.
KFD	Discriminant de Fisher non linéaire.
KPCA	Analyse en Composantes Principales non Linéaire.
LOO	«Leave-One-Out».
$\mathbb{R}$	Espace des réels.
x	Vecteur d'entrée arbitraire.
y	Sortie désirée bipolaire associée au vecteur x.
t	Sortie désirée binaire associée au vecteur x.
$P(x, y)$	Probabilité d'observation du couple (x, y).
l	Nombre d'exemples d'apprentissage.
f	Fonction de décision du SVM.
x_i	Vecteur d'entrée d'indice i.

α_i	Paramètre du SVM associé au vecteur x_i.
α	Ensemble des multiplicateurs α_i.
$R(\alpha)$	Risque réel.
$R_{emp}(\alpha)$	Risque empirique.
F	Espace augmenté.
N_F	Dimension de l'espace augmenté F.
$\Phi(x)$	Image de x dans l'espace augmenté.
h	Capacité du classifieur.
b	Paramètre de biais dans le SVM.
$k(.,.)$	Noyau de Mercer.
σ	Paramètre d'échelle du RBF et de KMOD.
γ	Paramètre d'écrasement de KMOD.
w	Vecteur orthogonal à l'hyper-plan de séparation du SVM.
R	Rayon de la plus petite hyper-sphère englobant les données dans F.
z	Observation dans l'espace F.
$L(w,b,\alpha)$	Lagrangien primaire du SVM.
$W(\alpha)$	Lagrangien dual du SVM.
ξ_i	Variable ressort associée à l'observation x_i.
C	Paramètre de compromis du SVM.
$\sharp sv$	Nombre de vecteurs de support.
ν	Paramètre de contrôle du nombre de vecteurs de support dans le ν-SVM.
β_i	Paramètre de l'hyper-sphère associé au vecteur x_i.

KKT	Conditions de «Karush-Kuhn-Tucker».
SMO	Optimisation par minimisation séquentielle.
Ω	Espace de partitionnement des classes.
A	Constante multiplicative de la sigmoïde.
B	Constante additive de la sigmoïde.
E_{cv}	Erreur de validation croisée.
PE	«Predicted error».
GPE	«Generalized predicted error».
Ψ	Fonction échelon.
θ_i	Hyper-paramètre particulier du SVM.
$\underline{\theta}$	Vecteur des hyper-paramètres du SVM.
$\underline{\theta}_0$	Vecteur des hyper-paramètres initial du SVM.
p_i	Probabilité à posteriori de l'observation x_i.
E_i	Erreur empirique associée à l'observation x_i.
E	Erreur empirique sur l'ensemble de validation.
$\underline{H}$	Matrice de Grahm-schmidt modifiée.
LS SVM	«Least square SVM».
O_i	Probabilité à posteriori de la classe i dans l'approche un-contre-un.
θ_{ij}	Hyper-paramètre associé au classifieur (i,j) dans l'approche un-contre-un.
LSM	«Line search minimization».

CHAPITRE 1

INTRODUCTION

La variabilité de l'écriture manuscrite permet de confronter les algorithmes de classification et d'apprentissage à des problèmes difficiles et réalistes. Les réseaux de neurones ont montré des résultats remarquables dans ce domaine. Mais la nécessité de performances élevées dans des applications réelles ont poussé la recherche vers des modèles de classification de plus en plus complexes. Ainsi, depuis le neurone formel jusqu'au Lenet5 de ATT [55], plusieurs générations de machines d'apprentissage ont vu le jour dans le but de classifier, de catégoriser ou de prédire des structures particulières dans les données. La classification de caractères constituait, par ailleurs, la principale application des machines d'apprentissage dont les différents travaux ont permis une nouvelle appréhension des modèles de classification. Dans le domaine de la reconnaissance du manuscrit, les premiers systèmes dits intelligents reconnaissaient déjà des caractères imprimés à l'aide du perceptron de Rosenblatt [83]. Par la suite, le besoin d'automatisation massive a donné lieu à toute une multitude d'applications dont la lecture de chèques bancaires, des adresses postales, des documents imprimés, etc.

Jusqu'à très récemment, le perceptron multicouche (PMC) constituait le modèle de prédilection pour traiter diverses tâches d'apprentissage supervisé. Ce modèle est facile à construire et à entraîner, et a été efficacement adopté dans beaucoup de systèmes de reconnaissance de caractères.

En 1979, Vapnik a mis au point le principe de la minimisation du risque structurel qui évite le sur-apprentissage des données à la convergence, contrairement au risque empirique qui représente un estimé optimiste de l'erreur et que minimise le PMC avec l'algorithme de rétro-propagation de gradient [119]. Depuis, le SVM (de l'anglais «Support Vector Machine») n'a cessé de susciter l'intérêt de plusieurs communautés de chercheurs

de différents domaines d'expertise. Ainsi, Cortes et al. dans [27], Scholkopf et al. dans [88], et Burges et al. dans [22] ont appliqué le SVM à la reconnaissance de chiffres manuscrits isolés. Blanz et al. dans [17] ont utilisé le SVM pour la classification d'objets 2D de vues différentes. Schmidt et al. dans [86] ont exploré la tâche de reconnaissance d'orateur. Osuna et al. ont développé un système de reconnaissance de visages [74]. Dans la plupart des cas, la performance du SVM égale ou dépasse celle des classifieurs classiques.

1.1 Motivation

Le SVM est un modèle discriminant qui tente de minimiser les erreurs d'apprentissage tout en maximisant la marge séparant les données des classes. La maximisation de la marge est une méthode de régularisation qui réduit la complexité du classifieur. Elle sert à pénaliser les paramètres du modèles de la même façon que la méthode du «weight decay» qui altère les poids de grande amplitude dans un PMC [43]. La pénalisation des paramètres du SVM dans le but de maximiser la marge est une méthode de sélection de modèle implicite à l'apprentissage. Ce processus produit un ensemble réduit de prototypes faisant partie de l'ensemble d'apprentissage qu'on appelle communément vecteurs de support. Son comportement est, par ailleurs, conditionné par le type du noyau utilisé et par les valeurs octroyées à ses paramètres. Le noyau d'un SVM est une fonction symétrique défini-positive qui permet de projeter les données dans un espace transformé de grande dimension dans lequel s'opère plus facilement la séparation des classes, de la même façon que les neurones cachés d'un PMC permettent de projeter les données entrées dans un espace de représentation dans lequel les poids de la couche de sortie définissent des fonction discriminantes linéaires des classes. Les valeurs des paramètres de noyaux affectent directement la complexité de la frontière de décision du classifieur. Aussi, ses valeurs influencent le nombre de vecteurs de support du classifieur.

De nombreux travaux ont démontré la supériorité du SVM sur les méthodes discriminantes classiques telles que le PMC, le discriminant de Fisher, le réseau RBF, etc. Des versions

modifiées du SVM permettent les meilleures performances sur plusieurs bases de données standards [55]. Sa robustesse vis à vis de la dimensionalité des données et son pouvoir accru de généralisation, font que le SVM est nettement plus avantageux.

Le SVM compte principalement deux variantes : SVM L1 et SVM L2. La première applique une pénalisation quasi-linéaire sur l'erreur d'apprentissage alors que la deuxième opère une pénalisation quadratique de l'erreur. Le SVM L2, de par sa règle d'inférence quadratique, est sensible aux données marginales dont l'effet peut biaiser notablement la solution du SVM surtout en l'absence d'une quantité suffisante de données. Aussi, dans un problème fortement débalancé (distribution a priori des classes non égales), l'influence des exemples de la classe majoritaire est magnifié au risque de négliger les exemples de la classe minoritaire. Le terme 'amnésie' est parfois utilisé pour désigner ce phénomène. Cependant, le SVM L1 est moins sensible aux données marginales. Aussi, en présence de classes fortement débalancées, sa solution est plus robuste. Dans cette thèse, nous avons opté pour cette variante du SVM [1].

Par ailleurs, le SVM est un classifieur binaire, qui ne traite habituellement que des données étiquetées par rapport à deux classes d'appartenance. Donc, traiter des données multiclasses comme c'est le cas en reconnaissance de chiffres, requiert la combinaison d'un ensemble de SVMs, chacun se spécialisant en une partie du problème. Parmi les schémas de combinaison les plus utilisés, on citera l'approche un-contre-tous et l'approche un-contre-un. La première consiste à entraîner chacun des classifieurs pour séparer une classe du restant des classes. La deuxième consiste à entraîner les SVMs afin d'obtenir toutes les frontières de décision séparant les classes une à une. Il en existe K(K-1)/2 pour un problème à K classes. D'autres schémas de décomposition permettent de considérer un plus grand nombre de classifieurs à combiner. Dietterich et al. montrent que la performance de l'ensemble dépend de son pouvoir à corriger les éventuelles erreurs de vote produits

[1] Excepté pour le chapitre 7 où nous effectuons quelques simulations avec la dimension VC et GACV L2 sur le SVM L2.

par une partie des classifieurs [29]. D'où l'appellation «Code Correcteurs d'Erreurs» qui désigne ce schéma de décomposition d'un problème multiclasse. Allawein et al. lient la performance de la méthode au nombre de décompositions et au degré de corrélation (indépendance) des classifieurs ainsi constitués [4].

Plusieurs approches ont été tentées pour formaliser mathématiquement dans un cadre unifié l'effet des hyper-paramètres sur le pouvoir de généralisation du SVM. Ainsi Sollich dans [100] présente une étude assez intéressante montrant le lien entre l'apprentissage du SVM et le formalisme de l'apprentissage Bayesien introduit par MacKay une décennie plutôt [61]. L'auteur ne présente toutefois pas de procédure automatique pour la sélection de modèle.

Scholkopf et Vapnik dans [87, 116] proposèrent de sélectionner le degré du noyau polynomial du SVM avec la dimension VC. Cette dernière, connue aussi sous le nom de dimension de Vapnik-Chernovenkis, traduit le degré de complexité du classifieur. Cette mesure doit idéalement être minime. Chapelle et al. présentent une méthode automatique de sélection de modèle basée sur la minimisation de la dimension VC et du «Span Bound» [24]. La dimension VC est cependant un critère qui n'est valable que sous l'hypothèse que les données sont séparables.

Dans le but de choisir adéquatement les hyper-paramètres du SVM, une sélection manuelle des valeurs des hyper-paramètres du classifieur peut être entreprise. Dans ce cas, il est important de pouvoir définir au préalable une zone d'intérêt dans laquelle plusieurs valeurs seront testées. Sans aucune information a priori, celle-ci n'est pas facile à trouver. Aussi, considérer une grande zone d'intérêt est très prohibitif en temps de calcul. En outre, il est nécessaire d'échantillonner cet intervalle de façon à assurer la plus grande résolution. Cette procédure devient exagérément complexe dès lors qu'il existe plus qu'un hyper-paramètre à considérer. Il est donc très important de disposer d'une procédure automatique de sélection de modèle qui s'affranchit de la sélection manuelle des valeurs des hyper-

paramètres. Cette procédure, peut être intégrée dans le processus d'apprentissage et perçue comme une procédure de calibrage du classifieur.

1.2 Propositions

Dans le cadre de cette thèse, nous proposons un nouveau critère de sélection automatique de modèle basé sur la minimisation d'un estimé de l'erreur de généralisation. Cette dernière que nous appelons «Erreur Empirique» représente une approximation de la probabilité d'erreur, et est estimée sur la base d'un ensemble de validation disjoint de l'ensemble d'apprentissage. L'algorithme est capable d'adapter les paramètres de tout type de noyau. L'approche est basée sur la minimisation automatique des valeurs du critère en calculant sa sensibilité par rapport aux hyper-paramètres. La méthodologie adoptée optimise les paramètres du noyau et minimise la complexité du classifieur.

Pour des données multiclasses, nous proposons deux schémas de sélection de modèles qui exploitent différemment les étiquettes des données. Le premier consiste à optimiser individuellement les classifieurs impliqués tel que nous faisons pour un problème de classification strictement binaire. Dans le deuxième, nous proposons une dérivation originale du problème qui consiste à considérer la totalité des hyper-paramètres de l'ensemble de SVMs simultanément et d'optimiser leurs valeurs de façon concourante. Le critère proposé est une estimation de la probabilité de l'erreur basée sur l'erreur quadratique moyenne. Cette approche est appelé *«méthode globale»* parce qu'elle cherche un optimum global des hyper-paramètres de l'ensemble de SVMs. Par opposition, la première est dite *«méthode locale»* parce qu'elle optimise localement les classifieurs.

Par ailleurs, nous proposons un nouveau modèle de noyau SVM pour la classification. Le noyau baptisé 'KMOD' (de l'anglais «Kernel with Moderate Decreasing») est une fonction radiale et symétrique. Il procure un comportement qui le distingue du reste des noyaux. En effet, nous analysons la notion de voisinage et de corrélation dans l'espace

augmenté de classification pour mettre en relief les avantages de KMOD par rapport à d'autres noyaux classiques tels que le RBF ou le noyau Laplacien [5, 8].

1.3 Organisation de la thèse

Ce manuscrit est organisé en une partie bibliographie, une partie méthodologie et une partie d'expérimentation.

Bibliographie La partie bibliographique est organisée en trois chapitres. Dans le chapitre 2, nous rapportons une revue de la littérature sur la reconnaissance de l'écriture depuis l'avènement des premiers systèmes de reconnaissance optique de caractères jusqu'aux derniers développements dans la reconnaissance en ligne et hors-ligne du manuscrit. On trouvera aussi une classification des principales méthodes d'extraction de caractéristiques reprise de Laaksonen [52], ainsi qu'une brève revue des principales méthodes discriminantes de classification. Dans le chapitre 3, on présentera une introduction à la théorie de l'apprentissage statistique dont dérive le principe du risque structurel. En second lieu, nous présenterons la théorie des machines à vecteurs de support après un bref rappel historique des travaux qui ont le plus influencé la théorie. En outre, on y trouvera une analyse des principaux algorithmes d'apprentissage du SVM. Dans la section 3.11 du même chapitre, nous présenterons différentes stratégies de combinaison et de normalisation des votes du SVM.

Enfin dans le chapitre 4, nous présenterons une revue de la littérature des principales méthodes de sélection de modèle. Nous mettrons en relief la forte connexion entre la régularisation des classifieurs via les hyper-paramètres et le principe du rasoir d'Occam établissant le compromis biais-variance de l'apprentissage statistique. On y trouvera quatre approches principales :

- approche par estimation de l'erreur de validation,
- approche algébrique,

- approche bayesienne,
- approche par minimisation du risque structurel.

Plus loin, nous présenterons des critères spécifiques à l'optimisation des hyper-paramètres du SVM. Entre autre, un emphase particulier est fourni pour expliquer quelques uns des critères analytiques les plus connus tels la borne rayon-marge dérivée de la dimension VC et le GACV.

Méthodologie

L'objectif de cette partie du travail est triple :

- proposer une méthodologie de sélection de modèle du SVM pour un problème de classification binaire,
- proposer un cadre unifié permettant la combinaison des ensembles de SVM en vue de traiter des données multiclasses,
- adapter la méthodologie de sélection de modèle du SVM aux problèmes multi-classes.

Dans le chapitre 5, nous abordons la problématique d'optimisation des hyper-paramètres du SVM sur des données binaires. Nous développons alors l'algorithme de minimisation de la dimension VC présenté par Chapelle et al. dans [24]. Dans un deuxième temps, nous présentons les inconvénients de la méthode qui requiert une procédure d'optimisation quadratique pour calculer l'hyper-sphère englobante des données.

Dans un deuxième temps, nous introduisons notre alternative qui consiste à pouvoir estimer la probabilité d'erreur sur des données indépendantes de l'ensemble d'apprentissage. Cette estimation est une statistique probabiliste qui permet d'optimiser les paramètres du noyau du SVM. Aussi, nous décrivons la procédure d'estimation de la probabilité à posteriori des observations à partir des sorties brutes du classifieur. Plus loin, nous présentons un schéma d'analyse de sensibilité qui permet de quantifier l'influence des hyper-paramètres

sur la probabilité d'erreur estimée. Ce schéma sera associé à une méthode de descente de gradient pour minimiser le critère.

Dans un troisième temps, nous considérons le GACV (de l'anglais «Generalized Approximate Cross Validation»), un critère analytique approximant une borne pessimiste de l'erreur de validation croisée «LOO». Il a été proposé par Wahba afin d'estimer les paramètres d'une SPLINE sur un problème de régression [125]. En 1996, Wahba et Lie proposèrent une version du critère étendue au cas de la classification avec le SVM. Son expression est une fonction linéaire des paramètres α_i du SVM et son calcul est effectif[2] une fois le classifieur entraîné. Pour des fins de comparaison, nous développons l'algorithme de minimisation automatique du GACV pour les deux variantes L1 et L2 du SVM.

Dans le chapitre 6, nous abordons le problème la sélection de modèle lorsque plusieurs SVM interagissent au sein d'un ensemble de classifieurs, chacun se spécialisant en une sous-tâche particulière de la classification. En effet, dans une stratégie un-contre-un comme celle adoptée dans notre système, la question de la stratégie de l'optimisation de l'ensemble émane naturellement. Nous proposons alors deux nouvelles méthodologies d'optimisation du SVM pour l'approche un-contre-un. La première baptisée «approche locale», considère l'optimisation individuelle des SVMs sans aucune interaction entre eux. Ainsi, l'optimisation des hyper-paramètres est réalisée de la même manière que pour un problème binaire, en ne considérant que les données correspondant au couplet de classes en question. La deuxième approche, baptisée «approche globale», permet l'optimisation conjointe de l'ensemble des paramètres de noyaux en minimisant une estimation de l'erreur multiclasse. Cette dernière est une mesure de vraisemblance (moins son logarithme) des données de validation. Ce schéma d'optimisation, permet d'inclure les données marginales (données n'appartenant pas au couple de classes considéré) dans le processus d'optimisation.

[2] Contrairement à la dimension VC qui nécessite l'estimation de l'hyper-sphère englobante

Expérimentation

Dans le chapitre 7, nous validons l'algorithme de minimisation de l'erreur empirique sur un problème de données binaires. La simulation y est effectuée en le comparant aux critères GACV et VC. Les expériences sont effectuées pour diverses valeurs initiales des hyper-paramètres et différentes valeurs du paramètre de régularisation C. Aussi, nous validons l'algorithme de minimisation automatique du GACV et démontrons la supériorité de l'estimé de l'erreur empirique sur la dimension VC.

Dans le chapitre 8, nous présentons les résultats d'expérimentation des méthodologies proposées sur des données multiclasse. Ainsi, en premier lieu, nous présentons les résultats de simulation sur un problème synthétique à trois classes. Il s'en suit une deuxième partie qui est consacrée aux expérimentations effectuées sur la base de données de chiffres manuscrits USPS. Pour celle-ci, nous présentons d'abord les résultats de sélection manuelle des hyper-paramètres en utilisant différents noyaux. Dans un deuxième temps, nous présentons les résultats d'expérimentation de la sélection automatique de paramètres. Enfin, les résultats d'expérimentation sur la base de données de chiffres indiens INDCENPARMI sont présentés. Aussi, on trouvera une description détaillée de la procédure d'extraction de caractéristiques utilisée.

«No more things should be presumed to exist than are absolutely necessary»,
William of Occam

CHAPITRE 2

RECONNAISSANCE DE CARACTÈRES MANUSCRITS

2.1 Introduction

La reconnaissance automatique du texte manuscrit et imprimé est un domaine qui a rapidement intéressé les chercheurs avant même l'avènement des premiers ordinateurs. À la base, l'intérêt pour la reconnaissance optique des caractères a commencé avec Tyurin en 1900, De Albe en l'an 1912 et Thomas en l'an 1926 qui ont tenté de mimer l'interprétation humaine de l'information visuelle. Mais, l'âge d'or de la discipline a débuté avec l'invention du premier ordinateur en l'an 1946 par Mauchly et Eckert. Quelques années plus tard, les premières expériences en reconnaissance de caractères ont pu être réalisées. Pendant les années soixante et soixante dix, les premiers systèmes de lecture automatique du texte imprimé ont vu le jour [10]. Toutefois, des systèmes fiables étaient restreints à quelques fontes seulement. Dans la même période, Lindgren dans [59], a introduit l'un des premiers systèmes de reconnaissance de l'écriture manuscrite. Les techniques de classification mises en oeuvre durant cette période font appel à des méthodes d'appariement syntaxique (la plupart des méthodes statistiques n'étant pas connues encore). Entre les années 1980 et 1990, les réseaux de neurones, découverts deux décennies avant, puis vite abandonnés, suscitent un vif intérêt de la communauté des chercheurs grâce à l'algorithme de rétro-propagation du gradient mis au point par Werbos [127], Lecun [54] et Rumelhart [85]. Ainsi, le perceptron multicouche a été rapidement reconnu comme le classifieur par excellence dans beaucoup de problèmes de reconnaissance de caractères. Un peu plus tard, nous avons vu naître aussi d'autres architectures plus complexes ou l'extraction de caractéristiques est intégrée dans le processus d'apprentissage du réseau comme dans Lenet 4 et Lenet 5 [55].

De nos jours, il existe un riche état de l'art présentant un excellent aperçu de l'état d'avancement du domaine [104, 106, 105, 75, 65, 40, 108].

Un système de reconnaissance de l'écriture doit idéalement, localiser, reconnaître et interpréter n'importe quel texte ou nombre écrit sur un support de qualité arbitrairement variable tel que des cartes, des formulaires, des agendas, des vieux manuscrits, etc. Ce type de systèmes aide à convertir l'information contenue dans les vieux manuscrits pour être enregistrée dans des bases de données que tout le monde peut interroger. Par la suite, on s'est vite aperçu de la difficulté de réaliser des systèmes ad hoc capables de reconnaître tout type de texte et on s'est tourné vers des systèmes dédiés, où l'on connaît a priori le style d'écriture que le système doit traiter. Par exemple, il existe des systèmes pour la lecture automatique des chèques bancaires, la reconnaissance du code et de l'adresse postale, la lecture des formulaire d'impôts, etc [101, 35].

Parmi les plus récentes plate-formes disposant d'un système de reconnaissance de l'écriture, nous trouvons le Palm et l'agenda électronique. Ces deux appareils regroupent une tablette à numériser et un programme procédant à la reconnaissance de l'écriture. De ce fait, son utilisation est plus attrayante puisqu'elle épargne à l'utilisateur le besoin de 'scanner' a priori son écriture. Ceci a remis la reconnaissance en ligne au centre d'intenses efforts de développement au sein de la communauté de l'écrit et du document. Parmi les utilisateurs potentiels de cette technologie, on trouve les établissements gouvernementaux tels que l'agence de douane et de l'impôt, la poste, ou alors les grandes corporations privées telles que les banques ou les éditeurs de magazines qui voudraient numériser leurs formulaires d'abonnement.

2.2 Reconnaissance de caractères

Les applications de la reconnaissance de caractères peuvent varier substantiellement selon la nature de l'écriture et son support de saisi. Dans cette étude, nous nous intéressons à la reconnaissance hors-ligne de chiffres manuscrits isolés. Les différentes applications de

la reconnaissance de caractères ne peuvent être dissociées de façon exclusive tant elles partagent plusieurs caractéristiques.

La reconnaissance de caractères, en effet, signifie le décodage de n'importe quel texte imprimé ou manuscrit, ou information symbolique incluant par exemple les formules mathématiques, le dessin technique, etc. De ce point de vue, le moyen utilisé pour transcrire l'information et la traiter est sans importance. Donc, la représentation des caractères n'est pas restreinte aux seules images comme c'est le cas dans la reconnaissance en-ligne. La figure 1 présente une classification des différents types d'écriture faite selon le média, le support de saisi et l'application considérés. La distinction entre le texte imprimé et le texte manuscrit bien qu'évidente n'est pas toutefois exclusive. Un système qui reconnaît du manuscrit doit être capable de reconnaître de l'imprimé. Aussi, certaines fontes imprimées miment l'écriture manuscrite. Mais, étant donné que les caractères manuscrits sont sujets à plus de bruit que ceux imprimés, la reconnaissance du manuscrit est considérée plus difficile que la reconnaissance de l'imprimé pour lequel des systèmes opérationnels commerciaux existent déjà.

La saisie des caractères manuscrits y est faite selon deux façons : en-ligne ou hors-ligne. En comparaison avec les systèmes hors-ligne, les systèmes en-ligne prennent en compte l'information chronologique des mouvement du bras du scripteur. Cette information additionnelle augmente la précision bien que souvent coûteuse en temps de calcul. La recherche sur les algorithmes de l'écriture en ligne est assez récente et il existe un format de données standard qui a été proposée par Guyon et al. [38].

Il existe une autre distinction à faire entre le texte (caractères) continu et les caractères isolés. Ces derniers sont présents spécialement dans les formulaires où la zone d'écriture est restreinte. Par contre, le texte continu nécessite une phase de segmentation préliminaire à la classification. Les systèmes traitant du texte imprimé considèrent que le texte est continu. Ainsi, les caractères continus forment des entités sémantiques de plus haut niveau

représentant des mots ou des nombres dont l'écriture manuscrite en est un cas particulier. La segmentation est alors bien plus complexe où l'on peut avoir recours à des méthodes de prédiction de séries temporelles comme les modèles de markov cachés (HMM) [31]. Néanmoins, le texte continu contenant des caractères pouvant se toucher mais n'affectant pas leurs caractères voisins, tel le script cursif, est sans doute un cas plus facile à traiter. Aussi, notons que les premiers systèmes ne traitaient que du texte imprimé avec certaines fontes dédiées aux OCRs. Plus tard, d'autres systèmes traitant une variété de fontes ont vu le jour.

Les systèmes modernes d'OCR peuvent reconnaître plusieurs fontes et s'adaptent à un nombre arbitraire de nouvelles fontes. Enfin, la dernière classification est valable aussi bien pour l'imprimé que pour le cursif et dissocie le texte contraint de celui qu'il ne l'est pas (Figure 1). Le premier est le cas quand le scripteur est appelé à écrire dans des zones prédéfinies. Le système dispose alors d'une information a priori lui permettant de pouvoir détecter et extraire l'information plus aisément. La reconnaissance de caractères isolés peut être perçue comme un cas spécial de cette branche. Le script non contraint peut par contre prendre n'importe quelle forme selon n'importe quelle direction et selon n'importe quel style. Ceci ramène au problème de l'analyse de documents qui constitue une branche à part du domaine. Dans l'analyse de documents, plusieurs types d'information sont extraits sans aucune connaissance a priori. En pratique, du texte est dit non contraint s'il contient plus qu'une ligne et lorsque le nombre de mots par ligne n'est pas connu. Aussi, les systèmes de reconnaissance doivent reconnaître des chiffres ou des symboles de ponctuation. On pourrait, par ailleurs, considérer les lettres de l'alphabet soit en majuscule ou en minuscule.

2.3 Extraction de caractéristiques

Dans cette section, nous présentons un aperçu des méthodes d'extraction de caractéristiques utilisées en reconnaissance de caractères manuscrits. En réalité, en plus de la taxo-

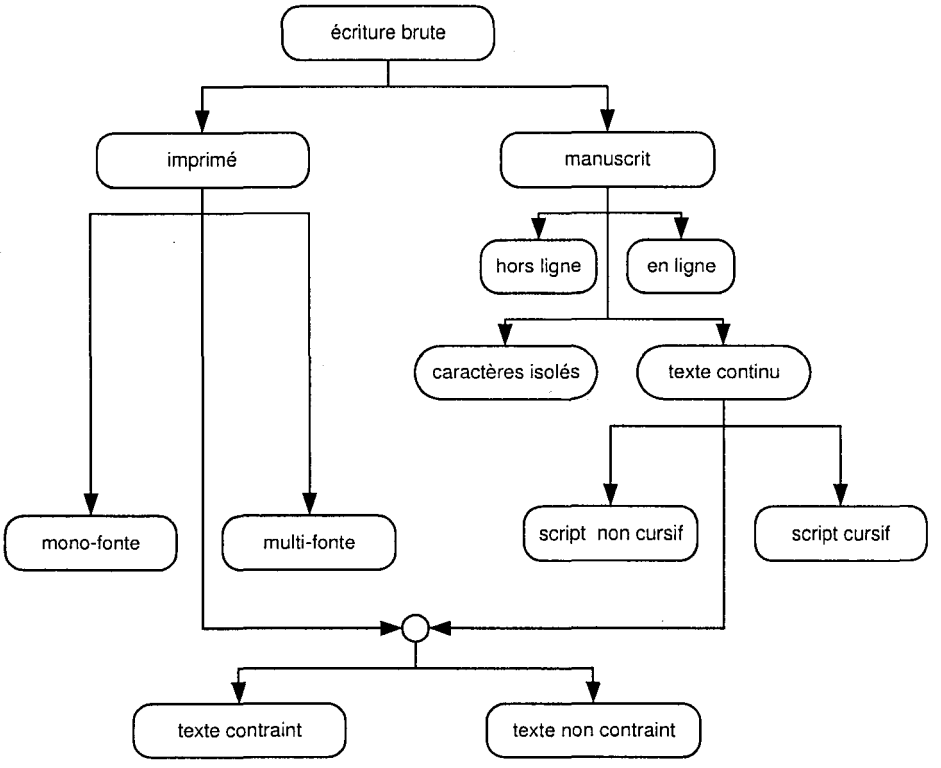


Figure 1 Dichotomie des types d'écriture

nomie présentée dans la figure 1, le domaine de la reconnaissance de caractères peut être décrit par la méthode de collection de données, des méthodes d'extraction de caractéristiques, des méthodes de classification ou du format de représentation des données. Trier et al. [113] ont réalisé une taxonomie des méthodes d'extraction selon le format de représentation de l'image qui est reprise par Laaksonen et que nous présentons dans tableau I [52].

Le tableau I montre quatre formats de représentation d'image arrangés en colonnes qui sont : image de gris, image binaire, le contour et le squelette. Pour chaque format, des méthodes d'extraction de caractéristiques raisonnables sont indiquées. Notons que le choix des caractéristiques et du format de représentation ne sont pas totalement dépendants ni tout à fait indépendants. Les méthodes d'extraction peuvent être distinguées selon deux classifications différentes décrites dans le tableau II. Cette taxonomie s'appuie sur la re-

Tableau I

Taxonomie des méthodes d'extraction de caractéristiques selon la représentation de l'image, d'après [52]

Caractéristiques	**Image de gris**	**Image binaire**	**Contour**	**Squelette**
Appariement	X	X		X
Motifs déformables	X			X
Transformation unitaire	X	X		
Transformation log-polaire	X	X		
Moments géométriques	X	X		
Moments de Zernik	X	X		
Ondelettes	X	X		
Algébriques	X	X		
Histogrammes de projection		X		
Masques	X	X		
Profil du contour			X	
Code de Freeman			X	
Spline			X	
Descripteurs de Fourrier			X	X
Description graphique				X
Discrets				X
Zonage	X	X	X	X

lation entre les caractéristiques et la méthode de classification utilisées. Ainsi, deux catégories de classifieurs sont considérées : structurel et statistique. Les caractéristiques sont organisées en lignes selon deux classes : heuristique et systématique.

Tableau II

Taxonomie des caractéristiques selon la méthode de classification

	structurel	**statistique**
heuristique	• caractéristiques discrètes	• zonage
systématique	• code de Freeman • spline • graphe	• volumétrique • profil du contour • descripteurs de Fourrier

2.4 Classification par discrimination

La classification d'une forme donnée se fait généralement selon trois étapes :

- extraction de caractéristiques,
- évaluation de la fonction discriminante,
- prise de décision.

Nous avons vu plus haut qu'il existe une multitude de méthodologies permettant de développer une représentation adéquate du tracé de l'écriture (Tableaux I et II). Ses caractéristiques, jumelées à un classifieur efficient, permettent de prédire l'appartenance d'une forme donnée quelque soit la complexité et le nombre de classes du problème. La fonction discriminante requiert un ensemble de données sur lequel ses paramètres seront appris. La précision de la fonction discriminante estimée est proportionnelle à la taille des données d'apprentissage. Enfin, la prise de décision est le processus d'émettre l'hypothèse la plus vraisemblable quant à l'appartenance d'une observation de test particulière.

Dans ce qui suit, nous présentons une brève revue de littérature de quelques modèles discriminants parmi les plus connus pour la reconnaissance de caractères manuscrits. On y trouvera aussi une introduction à la théorie de la décision dans le prochain paragraphe.

2.4.1 Décision Bayesienne

Dans une approche par modélisation, on cherche un modèle de production $p(x|c_i)$, qui donne pour chaque classe de formes c_i, la distribution des données qui lui sont associées. Dans une approche par discrimination, on cherche à approximer la distribution $p(c_i|x)$ qui représente la probabilité à posteriori des données étant donné la classe c_i. En pratique, cette information n'est pas toujours fournie.

La règle de décision bayesienne est une théorie clé en classification qui permet d'estimer la probabilité à posteriori à partir de la probabilité conditionnelle et d'émettre un vote

d'appartenance de la forme traitée. Elle s'écrit :

$$p(c_i|x) = \frac{p(x|c_i)p(c_i)}{p(x)}. \tag{2.1}$$

Pour K classes, la décision bayesienne cherche la classe c_i qui maximise la probabilité à posteriori. Elle s'écrit :

$$c(x) = \arg\max_i p(c_i|x)$$

Le terme $p(c_i)$ représente la probabilité a priori de la classe c_i. Il est particulièrement utile si les classes ne sont pas balancées dans l'échantillon de données considéré. La vraisemblance de l'observation $p(x)$, est une quantité constante qui peut être omise du processus de décision.

2.4.2 Perceptron multicouche

Les réseaux de neurones ont connu un essor important grâce à l'algorithme de rétro-propagation du gradient de l'erreur attribué à Werbos [127], Rumelhart [85] et Lecun [54]. Ce classifieur a trouvé application dans beaucoup de domaines tels que la reconnaissance de caractères, la reconnaissance de visages, la classification d'expressions de gènes, etc. Il est capable d'inférer n'importe quelle fonction de décision non linéaire moyennant une seule couche de neurones cachées et des fonctions d'activation sigmoïdales [51].

L'apprentissage du perceptron multicouche (PMC) est réalisé en minimisant une fonction de coût quadratique de l'erreur. Elle s'écrit :

$$E = \frac{1}{2l}\sum_{i=1}^{l}\sum_{k=1}^{r}(f_k(x_i) - t_i)^2,$$

r et l étant respectivement le nombre de classes et la taille des données considérées.

Lorsque le bruit des données est de nature gaussienne, il est démontré que les sorties du PMC estiment les probabilités à posteriori des classes [15].

Pendant la phase d'apprentissage, ce classifieur est assez rapide à entraîner. En test, il présente un temps de réponse qui dépend de sa complexité. Une architecture réduite est beaucoup plus rapide en test. Cependant, il n'existe pas de méthode permettant de choisir une taille adéquate du réseau de neurones. Cette limitation importante altère le pouvoir de généralisation du classifieur qui peut subir l'effet du sur-apprentissage. Ce phénomène prend une ampleur dangereuse lorsque le rapport du nombre de connexions au nombre de données est important. Aussi, le nombre de paramètres du PMC est proportionnel au nombre de caractéristiques. Ainsi, la dimensionalité des données peut dégrader grandement la performance du classifieur.

La théorie de régularisation [111, 78] a permis de proposer un premier cadre théorique pour gérer le problème du sur-apprentissage. La technique du "weight decay" est une méthode qui permet de réduire l'effet du sur-apprentissage en pénalisant une partie des paramètres du classifieur [43]. Elle constitue, sans doute, l'une des méthodes de régularisation les plus utilisées. Aussi, divers critères algébriques ont été proposés pour choisir a priori le nombre de paramètres du PMC ou pour élaguer une partie du réseau. La plupart demeurent toutefois prohibitifs en temps de calcul et sensibles aux données du problème. Nous verrons dans le chapitre 3, que le SVM réduit automatiquement le nombre de ses paramètres en maximisant la marge de séparation entre les données des classes. Celle-ci est une fonction de régularisation implicite à l'apprentissage qui a pour effet de produire un modèle clairsemé.

2.4.3 Réseaux de fonctions à base radiale

Les réseaux de fonctions à base radiale dits (de l'anglais «Radial Basis Functions network» ou «RBF network») s'appuient sur une modélisation de type mixture de gaussiennes des fonctions de sortie. Chaque sortie de classe devient une combinaison linéaire de probabilités issues de fonctions de distribution gaussiennes. Le nombre et les paramètres de celles-ci sont déterminés après catégorisation des données avec l'algorithme de K-MEANS ou ISODATA. Ensuite, un algorithme de rétro-propagation de gradient est exécuté afin de trouver les paramètres des perceptrons de la couche de sortie. Des variantes de l'algorithme permettent aussi d'adapter les paramètres des gaussiennes pendant l'apprentissage.

Au même titre que le SVM que nous verrons dans le prochain chapitre, sa fonction de décision peut être perçue comme une combinaison linéaire des mesures de similarité entre l'observation de test et un ensemble de prototypes (les centres des gaussiennes). La fonction de décision correspondant à la classe i s'écrit :

$$f_i(x) = Sig\left(\sum_{k=1}^{r} g_k(x) w_{ki} + w_{0i}\right),$$

g_k étant la fonction de probabilité de la $k^{ème}$ gaussienne, r étant le nombre de gaussiennes et Sig étant une fonction sigmoïde d'activation. Les paramètres $w_{ki}, k = 1, \ldots, r$ représentent les poids du perceptron de la classe i et w_{0i} représente le biais de la fonction de décision de la classe.

Ce réseau estime les probabilité conditionnelles des classes. Pour éviter le sur-apprentissage, ce réseau requiert habituellement une quantité plus importante de données que le PMC [15].

2.4.4 Réseau à convolution

Ces réseaux dits "Space Displacement Neural Network" (SDNN) sont basées sur le principe de partage de poids pour restreindre le nombre de paramètres du réseau. Les poids de ce dernier sont répartis entre plusieurs couches. Chaque couche de poids extrait des primitives de haut niveau de plus en plus complexes. La première est une rétine qui capture des primitives de bas niveau. Le réseau contourne ainsi l'utilisation de caractéristiques en traitant les valeurs de pixels brutes. La structure du réseau est cependant choisie manuellement selon des considérations empiriques [55].

Il est possible d'appliquer le réseau sur une image complète de mot ou de montant sans segmentation préalable du tracé. Le réseau montre une forte robustesse à certains bruits. L'apprentissage des poids du réseau est réalisé avec un algorithme de rétro-propagation de gradients avec une procédure d'accélération exploitant l'information de second ordre.

2.5 Conclusion

Dans ce chapitre, nous avons présenté un aperçu de la problématique de reconnaissance de caractères, l'application ciblée par ce travail.

En premier lieu, nous avons tenu à présenter un historique des principaux apports ayant influencés le développement du domaine. En outre, nous avons présenté une dichotomie des types d'écriture et de leurs traitements. Deux taxonomies des principales méthodes d'extraction de caractéristiques sont aussi fournies.

Dans la dernière partie du chapitre, nous rappelons les principes de la théorie de la décision et analysons quelques unes des principales méthodes discriminantes utilisées pour la reconnaissance de caractères telles que le PMC, le réseau RBF et le SDNN. Bien que performantes sur beaucoup de problèmes, leur principale lacune est la difficulté d'adapter adéquatement l'architecture du classifieur (réseau de poids) au problème de classification

en main. L'autre maillon faible des méthodes, est leur sensibilité à la dimensionalité des données.

Comme nous allons le voir dans le chapitre 3, le SVM procure un cadre théorique qui résout le problème du choix du nombre de paramètres puisque celui se fait automatiquement. L'efficacité du modèle pour la classification de données de grande dimension telles que du texte sur un très grand dictionnaire de mots a été établie par des travaux comme ceux de Joachims et al. [48, 49]. La robustesse et l'efficacité du SVM à traiter des données de nature et de complexité arbitraire a été établie sur beaucoup d'applications de reconnaissance de formes [56].

CHAPITRE 3

MACHINES À VECTEURS DE SUPPORT : ÉTAT DE L'ART

3.1 Introduction

L'origine des machines à vecteurs de support (SVM) remonte à 1975 lorsque Vapnik et Chervonenkis proposèrent le principe du risque structurel et la dimension VC pour caractériser la capacité d'une machine d'apprentissage . A cette époque, ce principe n'a pas trouvé place et il n'existait pas encore un modèle de classification solidement appréhendé pour être utilisable. Il a fallu attendre jusqu'à l'an 1982 pour que Vapnik propose un premier classificateur basé sur la minimisation du risque structurel baptisé SVM [120]. Ce modèle était toutefois linéaire et l'on ne connaissait pas encore le moyen d'induire des frontières de décision non linéaires. En 1992, Boser et al. proposent d'introduire des noyaux non-linéaires pour étendre le SVM au cas non-linéaire [18]. En 1995, Cortes et al. proposent une version régularisée du SVM qui tolère les erreurs d'apprentissage tout en les pénalisant [27].

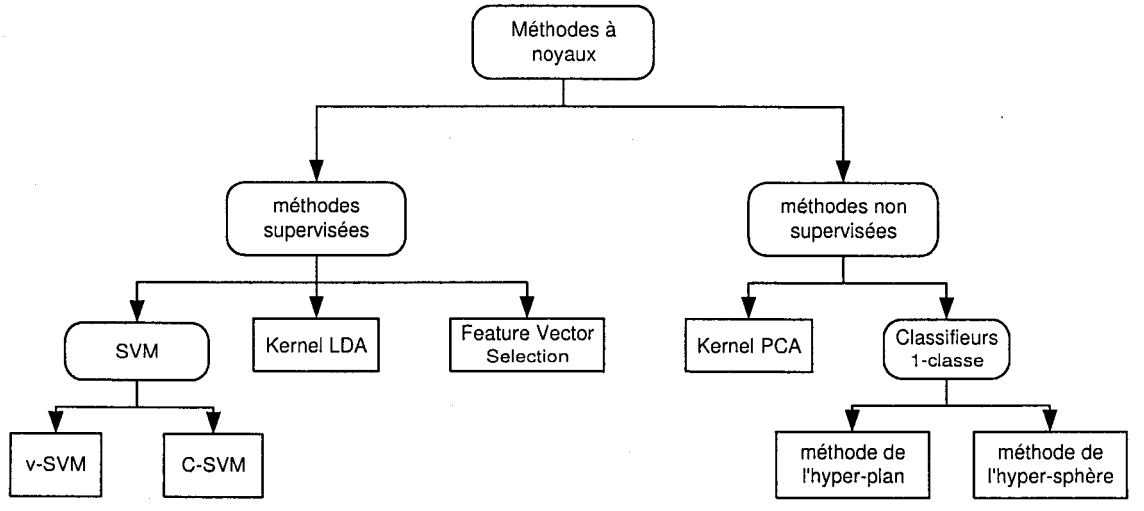


Figure 2 Arbre de classification des méthodes d'apprentissage à base de noyaux

Depuis, les SVMs (le pluriel est utilisé pour désigner les différentes variantes du SVM) n'ont cessé de susciter l'intérêt de plusieurs communautés de chercheurs de différents domaines d'expertise. Par exemple, Cortes et al. dans [27], Scholkopf et al. [88], et Burges et al. [22] ont appliqué les SVM à la reconnaissance de chiffres manuscrits isolés, Blanz et al. [17] ont expérimenté le SVM sur des objets 2D de vues différentes. Schmidt et al. [86] ont exploré la tâche de reconnaissance d'orateur. Osuna et al. ont traité de la reconnaissance d'images de visages [74]. Dans la plupart des cas, la performance du SVM égale ou dépasse celle de modèles classiques déjà établis. L'estimation de densité de probabilité [128] et la décomposition ANOVA [102] ont été aussi explorées. D'autres auteurs ont aussi étudié l'apport de la connaissance a priori pour ce type de classificateurs. Ainsi, le SVM virtuel de Burges et al. améliorent la généralisation en appliquant un bruit spécifique à l'ensemble de vecteurs de support [22].

Smola et al. ainsi que Wahba ont mis en évidence la ressemblance entre le SVM et la théorie de régularisation [96, 37, 122]. Ils ont démontré qu'associer un noyau particulier à un SVM revient à considérer une pénalisation différente de l'erreur d'apprentissage en maximisant la marge. Ce qui nous permet de dire que la maximisation de la marge dans l'espace augmenté est une forme de régularisation de l'apprentissage. Dès lors, le SVM permet de répondre à deux problèmes centraux de la théorie de l'apprentissage statistique que sont :

- le contrôle de la capacité du classifieur,
- le sur-apprentissage des données.

Dans le paragraphe 3.5.1, nous définissons les hyper-plans de séparation et éclaircissons le lien entre la notion de capacité et la marge de séparation entre deux classes. La notion de marge optimale est par la suite abordée davantage dans le paragraphe 3.5.2. La dérivation de l'algorithme du SVM est ensuite établie en deux étapes. Dans le paragraphe 3.5.3, l'algorithme est adapté au moyen d'une heuristique particulière au cas de données non

séparables. Enfin, le cas non-linéaire est traité dans le paragraphe 3.5.4. La figure 2 montre une classification des différentes méthodes à noyaux.

3.2 Risque structurel

Dans cette section, nous rappelons quelques éléments essentiels de la théorie de l'apprentissage statistique. Nous introduisons par la même occasion le principe du risque structurel que minimise le SVM [115].

Considérons un problème de classification à deux classes et soit les paires de données étiquetées :

$$(x_1, y_1), \ldots, (x_l, y_l) \in \mathbb{R}^N \times \{\pm 1\}, \tag{3.1}$$

où x_i représente la i^{eme} observation de l'ensemble d'apprentissage et y_i son étiquette.

Et soit l'ensemble de fonctions f_α défini :

$$\{f_\alpha \ : \ \alpha \in \Lambda\}, f_\alpha \ : \ \mathbb{R}^N \rightarrow \{\pm 1\}, \tag{3.2}$$

Supposons aussi qu'il existe une distribution $P(x, y)$ des données dont on ignore le modèle. La tâche d'apprentissage de f_α qui approxime au mieux cette distribution consiste à minimiser le risque réel donné par :

$$R(\alpha) = \int \frac{1}{2} |f_\alpha(x) - y| dP(x, y). \tag{3.3}$$

Ne connaissant pas $P(x, y)$, il est difficile d'estimer le risque $R(\alpha)$. Il est possible toutefois de considérer une fonction de risque empirique de la forme :

$$R_{emp}(\alpha) = \frac{1}{l}\sum_{i=1}^{l}\frac{1}{2}|f_\alpha(x_i) - y_i|. \tag{3.4}$$

Si l'ensemble d'apprentissage est fini, la minimisation du risque empirique donné par l'équation 3.4 ne garantit pas un minimum pour le risque réel.

En 1979, Vapnik [119] a mis au point le principe de la minimisation du risque structurel qui évite le sur-apprentissage des données à la convergence de la procédure d'apprentissage. Le risque empirique, par ailleurs, représente une estimation assez optimiste du risque réel dont la minimisation ne garantit pas la convergence vers une solution acceptable.

En fait, si nous choisissons d'entraîner un PMC avec une méthode de descente de gradient quelconque, nous minimiserons l'erreur quadratique sur l'ensemble d'apprentissage en considérant un grand nombre d'époques à travers les données. Or, le modèle trouvé ainsi représente une solution biaisée à cause de l'erreur de variance dont souffre l'erreur quadratique. En effet, il est démontré qu'un minimum pour l'erreur de généralisation requiert un compromis entre l'erreur de biais et l'erreur de variance. La technique du 'early stopping' par exemple, est une des règles empiriques utilisées pour éviter cet inconvénient.

Le risque structurel constitue une borne supérieure de l'erreur de généralisation qui s'écrit :

$$R(\alpha) \leq R_{emp}(\alpha) + \phi(\frac{h}{l}, \frac{\log(\eta)}{l}) \tag{3.5}$$

pour $\alpha \in \Lambda$ et $l \geq h$ avec une probabilité d'au moins $1 - \eta$, et où ϕ est un terme de confidence donnée par :

$$\phi(\frac{h}{l}, \frac{\log(\eta)}{l}) = \sqrt{\frac{h(\log \frac{2l}{h} + 1) - \log(\frac{\eta}{4})}{l}} \tag{3.6}$$

Le paramètre h est appelé VC ou dimension de Vapnik-Chervonenkis. Il décrit la capacité de l'ensemble de fonctions solutions. Pour un problème de classification binaire, h est le nombre maximum de points séparables selon 2^h configurations par l'ensemble de fonctions solutions. La minimisation de cette borne consiste à opérer une régularisation de l'erreur empirique $R_{emp}(\alpha)$ via le terme de pénalité $\phi(\frac{h}{l}, \frac{\log(\eta)}{l})$. Donc un minimum est garanti en minimisant à la fois l'erreur empirique et le terme de régularisation.

Le terme *machine d'apprentissage* est utilisé pour désigner l'ensemble de fonctions de décision f_α dont la machine dispose, le principe d'induction afin d'approximer l'erreur de généralisation et l'algorithme mettant en oeuvre le principe d'induction.

La limite supérieure du risque réel (erreur de généralisation) définie dans l'équation 3.5 constitue un principe essentiel de la théorie des Machines à Vecteurs de Support qui nécessite encore quelques remarques pertinentes.

Selon l'équation 3.5, étant donné un ensemble d'apprentissage de taille l, il est possible de borner $R(\alpha)$ en minimisant la somme des quantités : $R_{emp}(\alpha)$ et $h(\{f_\alpha \ : \ \alpha \in \Lambda^{'}\})$, tel que $\Lambda^{'}$ représente un sous-ensemble de Λ. Par ailleurs, l'erreur empirique dépend de la solution particulière trouvée par la machine d'apprentissage, à savoir f_α, et peut être réduite en choisissant des valeurs appropriées pour les paramètres α_i composant α.

La capacité h (dimension VC) quant à elle, dépend de l'ensemble de fonctions $\{f_\alpha \ : \ \alpha \in \Lambda^{'}\}$ que la machine d'apprentissage peut inférer. Dans le but de contrôler h, il est possible de considérer des structures $S_n := \{f_\alpha \ : \ \alpha \in \Lambda_n\}$ de plus en plus complexes, telles

que :

$$S_1 \subset S_2 \subset \ldots \subset S_n \subset \ldots, \tag{3.7}$$

et dont les capacités respectives vérifient l'inégalité :

$$h_1 \leq h_2 \leq \ldots \leq h_n \leq \ldots. \tag{3.8}$$

Pour un ensemble observations $(x_1, y_1), \ldots, (x_l, y_l)$, la minimisation du risque structurel vise à choisir f_α parmi l'ensemble de fonctions $\{f_\alpha \ : \ \alpha \in \Lambda_n\}$ possibles, qui minimise le risque garanti (le risque garanti a été utilisé par Guyon et Boser pour désigner la limite supérieure de l'erreur de généralisation donnée dans l'équation 3.5).

Le processus de choisir le bon sous-ensemble de fonctions solutions revient à contrôler la complexité du classifieur en cherchant le meilleur compromis entre une faible erreur empirique et une complexité moindre. Notons enfin, que plusieurs travaux ont abouti à des résultats similaires quant à la nécessité d'un compromis entre l'erreur d'apprentissage et la complexité du modèle pour assurer une faible erreur de généralisation. Il en est ainsi avec la théorie de la régularisation [111], le *«Minimum Description Length»* [82, 51] et le dilemme Biais-Variance [33].

3.3 Espace augmenté

Soit l'observation $\mathbf{x} = (x_1, \cdots, x_N) \in \mathbb{R}^N$, et supposons que son information utile soit contenue dans l'ensemble des monômes d'ordre d des composantes x_j de $\mathbf{x}$. Dans ce cas, il est possible de considérer l'espace F des monômes d'ordre d comme espace de caractérisation. En reconnaissance d'images, ceci équivaut à calculer des produits de valeurs de pixels.

Par exemple, dans l'espace d'entrée $\mathbb{R}^2$, l'ensemble des monômes d'ordre 2 constitue un vecteur de caractérisation de dimension 3, comme montré ci-dessous :

$$\Phi : \mathbb{R}^2 \rightarrow F = \mathbb{R}^3, \tag{3.9}$$

qui associe pour chaque observation $x = (x_1, x_2)$, une image de la forme :

$$(x_1, x_2) \mapsto (x_1^2, x_2^2, x_1 x_2).$$

Cette approche permet de considérer un degré de non-linéarité allant jusqu'à :

$$N_F = \frac{(N + d - 1)!}{d!(N - 1)!}, \tag{3.10}$$

où N_F représente la dimension de l'espace des monômes de degré d correspondant à l'espace d'entrée $\mathbb{R}^N$.

Une image de 16×16 pixels, par exemple, produit 10^{10} monomes d'ordre 5. Le calcul dans cet espace peut s'avérer fastidieux et très coûteux en complexité de calcul et en espace mémoire. Cependant, dans certaines conditions spécifiques, il est possible de considérer un espace augmenté Φ de très haute dimension sans calculer explicitement les données dans cet espace en utilisant des noyaux non-linéaires. Ces derniers permettent de projeter les données de l'espace d'entrée $\mathbb{R}^N$ vers un espace augmenté et manipuler uniquement les produits scalaires des images des points.

3.3.1 Noyau polynomial

Pour pouvoir représenter des produits scalaires de la forme $\Phi(x).\Phi(y)$ (voir équation 3.9), nous utiliserons la notation :

$$k(x, y) = \Phi(x).\Phi(y), \tag{3.11}$$

qui décrit la valeur du produit scalaire dans l'espace augmenté F.

Ce calcul ne nécessite pas le calcul des images $\Phi(x)$ et $\Phi(y)$ explicitement. Cette propriété a été utilisée par Boser, Guyon et Vapnik pour étendre l'algorithme de l'hyper-plan généralisé au SVM non-linéaire [18]. L'espace augmenté F est alors appelé espace de linéarisation (Figure 3).

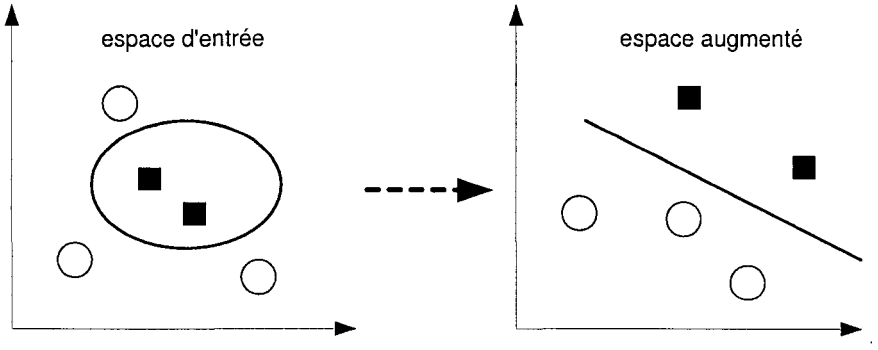


Figure 3 Transformation des données avec les noyaux de Mercer

En réalité, Aizerman et al. étaient les premiers à avoir élucider le concept de manipulation de produits scalaires via un noyau non-linéaire $k(x, y)$ [1].

Tout noyau décomposable selon la forme donnée dans l'équation 3.11 est un noyau Mercer [28]. On démontre que tout noyau continu, symétrique et semi-défini positif est un noyau de Mercer.

Si nous considérons l'exemple du paragraphe précédent avec $N = d = 2$, nous aurons alors une transformation de la forme :

$$C_2(x) \ : \ (x_1, x_2) \mapsto (x_1^2, x_2^2, x_1x_2, x_2x_1),$$

dont le produit scalaire $(C_2(x).C_2(y))$ peut s'écrire :

$$(C_2(x).C_2(y)) = x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 = (\mathbf{x}.\mathbf{y})^2.$$

Cet exemple simple montre qu'un noyau quadratique produit un espace augmenté implicite dans $\mathbb{R}^4$. Plus généralement, il est démontré qu'un noyau polynomial d'ordre d projette les données dans l'espace des monômes d'ordre d [78]. Une autre fonction Φ possible pour ce même noyau aurait pu être :

$$\Phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2). \tag{3.12}$$

Le tableau III montre quelques noyaux de Mercer classiques utilisés pour le SVM.

Tableau III

Quelques noyaux classiques

Noyau	**Formule**
Linéaire	$k(x, y) = x.y$
Sigmoïde	$k(x, y) = \tanh(a\mathbf{x}.\mathbf{y} + b)$
Polynomial	$k(x, y) = (ax.y + b)^d$
RBF	$k(x, y) = \exp(-\|x - y\|^2/\sigma^2)$
Laplace	$k(x, y) = \exp(-\|\gamma\|\|x - y\|)$

3.4 KMOD : un nouveau noyau pour la classification

En général, nous ne connaissons pas la fonction qui transforme l'espace d'origine en un espace augmenté plus grand. L'existence d'une telle fonction pour un noyau donné est assurée néanmoins par le théorème de Mercer [28]. Ces noyaux expriment un produit scalaire dans l'espace augmenté. Ils appartiennent essentiellement à deux familles :

1. noyaux de projection prenant la forme $k(x, y) = k(x.y)$,
2. noyaux de distance prenant forme $k(x, y) = k\left(||x - y||\right)$.

Dans la deuxième formulation, connaissant une estimation de la distance euclidienne entre deux points dans l'espace d'origine, nous obtenons leur degré de corrélation dans l'espace augmenté. Dans ce contexte, une question naturelle est : y a t-il un critère permettant de préférer un noyau à un autre ?

Dans le RBF (Figure 4), la corrélation des points voisins dans l'espace augmenté est contrôlée par un seul paramètre qui est l'écart type σ (Tableau III). Dans l'espace augmenté, les images des points très proches vont être hautement corrélées, alors que les images des points lointains vont être non corrélées. Ceci se traduit par une valeur du noyau égale à 1 pour une corrélation parfaite (distance nulle entre les points), et une corrélation nulle lorsque les points sont assez distants l'un de l'autre.

S'il était possible de contrôler davantage la corrélation des points dans l'espace augmenté, il serait possible d'influencer la séparabilité des classes dans le SVM puisque celle-ci dépend du produit scalaire

$$W.\Phi(x),$$

où $\Phi(x)$ est l'image de l'observation x par le noyau, et W est le vecteur orthogonal au plan de séparation du SVM (celui-ci dépend de l'ensemble des vecteurs de support et de leurs multiplicateurs α_i). En particulier, il serait souhaitable d'assurer un compromis qui permette de contrôler la dynamique de la fonction de corrélation tout en préservant

l'information sur le voisinage lointain. Cette propriété donnerait à l'algorithme le moyen de prendre en compte toute l'information de voisinage.

KMOD est un noyau que nous avons introduit dans [5, 8] et qui a l'avantage de présenter ce compromis. Son expression est :

$$k(x,y) = a\left(\exp(\frac{\gamma^2}{\|x-y\|^2+\sigma^2}) - 1\right). \tag{3.13}$$

Le paramètre a est une constante de normalisation de la forme :

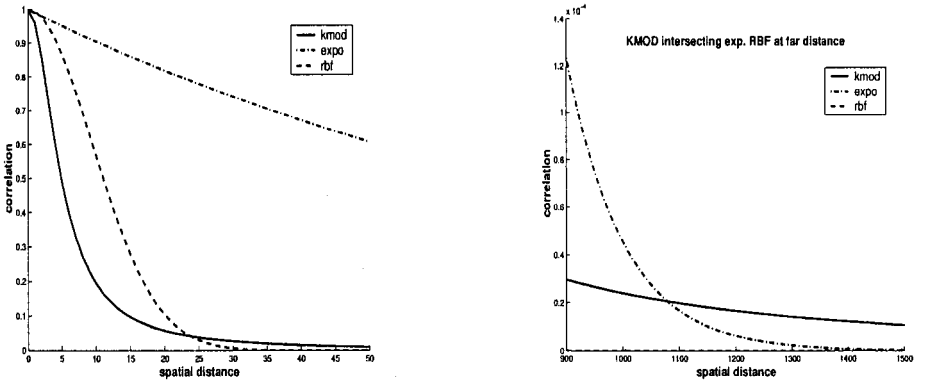
$$a = \frac{1}{\exp(\frac{\gamma^2}{\sigma^2}) - 1}.$$

Les variables γ et σ contrôlent l'écrasement du noyau. En particulier, σ est un paramètre d'échelle qui définit une fonction porte autour de zéro et γ contrôle l'écrasement autour de zéro. Le terme -1 dans l'expression garantie que KMOD converge vers zéro à l'infini. En particulier, le noyau satisfait deux propriétés :

- un écrasement modéré vers l'infini,
- une vitesse d'écrasement qu'on peut contrôler autour de zéro.

Dans la figure 4-a, nous constatons que KMOD maintient une vitesse d'écrasement autour de zéro plus rapide que celle du RBF. Cette vitesse est néanmoins plus modérée au-delà d'un certain voisinage. Par ailleurs, dans la figure 4-b, nous montrons que pour un voisinage lointain le noyau RBF se confine à zéro. Cependant, KMOD décroît plus lentement et préserve ainsi la similarité entre les images de ses points.

Le tableau IV présente une comparaison des résultats de classification du SVM avec KMOD, RBF et d'autres classifieurs sur la base de données «Breast Cancer» de UCI [16]. Nous constatons que KMOD fournit la meilleure performance.



(a) corrélation dans le voisinage proche (b) corrélation dans le voisinage lointain

Figure 4 Variation de la corrélation pour les noyaux KMOD, RBF et exp. RBF (Laplace)

Tableau IV

Performance de $KMOD$ (colonne 1) et autres classifieurs sur la base de données Breast Cancer [81, 8]. Le signe + indique si les valeurs sont significativement différentes de celle de $KMOD$ (test de student)

classifieur	SVM+KMOD	SVM+RBF	Réseau RBF	Adaboost	Adaboost Reg
erreur	25.4±4.4	26.0±4.7	+ 27.6±4.7	+ 30.4±4.7	26.5±4.5

Nous verrons dans la partie expérimentation, une étude plus complète des performances de KMOD sur d'autres bases de données de benchmark.

3.5 Formulation

Dans ce paragraphe, nous décrivons l'utilisation des machines à vecteurs de support pour la classification et dérivons la formulation du SVM linéaire. Nous y trouverons deux descriptions distinctes. La première traite le cas de données séparables. Une version modifiée permet, par ailleurs, de considérer des données non séparables. L'extension au cas non-linéaire est décrite plus loin.

Soit donc le problème de classification binaire défini par l observations $(x_1, y_1), \ldots, (x_l, y_l)$, tirées de manière indépendante d'une même distribution inconnue. Chacune des données $x_i, \forall i = 1, \ldots, l$, représente un vecteur de caractéristiques dans $\mathbb{R}^N$ de dimension N. Les variables $y_i = \{+1, -1\}, \forall i = 1, \ldots, l$ représentent les classes d'appartenance correspondant aux données x_i.

Il s'agit d'estimer la fonction de décision $f(x)$ qui approxime au mieux les exemples d'apprentissage, telle que $y_i = f(x_i)$ pour toute donnée x_i. Aussi, définissons la fonction de coût associée à l'erreur de classification comme suit :

$$E(y, f(x)) = \begin{cases} 0 \; si \; y = f(x) \\ 1 \; sinon \end{cases} \tag{3.14}$$

3.5.1 Hyper-plans de séparation

Nous avons énoncé dans l'équation 3.7 que des structures imbriquées S_n d'espaces de fonctions mettent en oeuvre le principe de la minimisation du risque structurel si un compromis entre la complexité du modèle et une faible erreur d'apprentissage est trouvé. L'algorithme de construction du SVM considère des hyper-plans de séparation entre les classes à discriminer.

Soit alors $z_1, \ldots, z_l \in F$ les images de $x_1, \ldots, x_l \in \mathbb{R}^N$ dans l'espace augmenté F.

L'hyper-plan de séparation dans l'espace F peut s'écrire :

$$\{z \in F : (w_0.z) + b_0) = 0\}, \tag{3.15}$$

où $(w, b) \in F \times \mathbb{R}$.

Cette formulation permet toujours de choisir des surfaces dont le couple (w, b) est un multiple de (w_0, b_0) de l'équation 3.15. Cependant, il est d'usage de considérer une formulation canonique telle que :

$$min_{i=1,\ldots,l}|(w.z_i) + b)| = 1, \tag{3.16}$$

qui garantit que les points les plus proches de la surface de séparation aient une sortie de valeur absolue égale à 1. Il est facile de démontrer, dans ce cas, que la distance séparant les exemples de la classe positive et les exemples de la classe négative les plus proches du plan de séparation (appelée aussi marge de séparation) est égale à :

$$D = \frac{2}{\|w\|}. \tag{3.17}$$

Le principe de minimisation du risque structurel dans la théorie des machines à vecteurs de support découle du fait qu'il est démontré que la dimension VC associée à des fonctions de décision de la forme :

$$f_{w,b} = sgn((w.z) + b) \tag{3.18}$$

vérifie :

$$h < R^2A^2 + 1. \tag{3.19}$$

Dans l'équation 3.19, R est le rayon de la plus petite hyper-sphère englobant les données d'apprentissage $z_1, \ldots, z_l$, et A un scalaire tel que $A \geq 2/D$. Si $\|w\| > A$, la borne supérieure de la dimension VC devient $N_F + 1$, où N_F est la dimension effective de l'espace augmenté F.

Il est établi que pour $\|w\| \leq A$, il est possible d'obtenir des dimensions VC largement inférieures à N_F. Cette propriété permet de travailler dans des espace de très haute dimension avec une dimension VC minimale. En résumé, maximiser la limite inférieure de la marge D permet de minimiser la dimension VC de la machine d'apprentissage. Bref, nous cherchons à minimiser à la fois le nombre d'exemples d'apprentissage mal classifiés et la dimension VC en maximisant la marge.

3.5.2 Hyper-plans à marge optimale

Supposons que nous ayons un ensemble d'observations $(z_1, y_1), \ldots, (z_l, y_l)$, tel que $z_i \in F$ et $y_i \in \{\pm 1\}$.

Idéalement, nous cherchons à trouver la fonction de décision

$$f_{w,b} = sgn((w.z) + b)$$

telle que :

$$f_{w,b}(z_i) = y_i, \ i = 1, \ldots, l. \tag{3.20}$$

Si cette fonction existe (le cas non-séparable sera traité dans la prochaine section), l'inégalité suivante est vérifiée :

$$y_i((z_i.w) + b) \geq 1, i = 1, \ldots, l. \tag{3.21}$$

Le lagrangien primaire s'écrit alors

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i(y_i((z_i.w) + b) - 1). \tag{3.22}$$

Les variables α_i correspondent aux facteurs de Lagrange des contraintes définies dans l'équation 3.21. Au minimum de la fonction objective $L(w, b, \alpha)$, ses gradients par rapport à w et b deviennent :

$$\frac{\partial}{\partial b} L(w, b, \alpha) = 0, \tag{3.23}$$

et

$$\frac{\partial}{\partial w} L(w, b, \alpha) = 0. \tag{3.24}$$

Résoudre les équations 3.23 et 3.24 donne :

$$w = \sum_{i=1}^{l} \alpha_i y_i z_i \tag{3.25}$$

avec

$$\alpha_i [y_i((z_i.w) + b) - 1] = 0, i = 1, \ldots, l. \tag{3.26}$$

En remplaçant w par son expression de l'équation 3.25 dans l'équation 3.22, et en prenant en compte la contrainte de l'équation 3.26, la fonction objective $L(w, b, \alpha)$ prend la forme d'un Lagrangien dual à maximiser fonction de α_i seulement. Il s'écrit :

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j}^{l} \alpha_i \alpha_j y_i y_j (z_i.z_j) \tag{3.27}$$

avec les contraintes :

$$\alpha_i \geq 0, \; i = 1, \ldots, l, \tag{3.28}$$

et

$$\sum_{i=1}^{l} \alpha_i y_i = 0. \tag{3.29}$$

La fonction de décision finale du SVM linéaire s'écrit :

$$f(z) = sgn\left(\sum_{i=1}^{l} \alpha_i y_i (z.z_i) + b\right). \tag{3.30}$$

Les données d'apprentissages z_i associées à des paramètres α_i non nuls sont appelées vecteurs de support. Pour une observation de test z quelconque, la valeur de $f(z)$ indique sa classe d'appartenance inférée par le SVM.

3.5.3 Hyper-plans à marge molle

Le problème d'optimisation quadratique énoncé dans l'équation 3.27 a une solution dans le cas de données séparables uniquement. Dans le cas contraire, les conditions de Tucker (KKT) ne sont jamais satisfaites.

Cortes et al. [27] utilisent une technique qui consiste à accepter des erreurs d'apprentissage tout en les pénalisant. Les auteurs ont introduit un paramètre de pénalisation C qui règle le degré de compromis désiré entre la séparabilité des classes et l'étanchéité du modèle aux erreurs d'apprentissage. Pour ce faire, redéfinissons la contrainte de l'équation 3.21 en considérant des variables :

$$\xi_i \geq 0,\ i = 1, \ldots, l,$$

associées aux données $z_i \in F, \forall i = 1, \ldots, l$ de l'ensemble d'apprentissage.

La variable ξ_i est la distance séparant z_i de la frontière de la marge qui est définie : $\xi_i = [1 - f(z_i)y_i]_+$, pour $i = 1, \ldots, l$. L'opérateur $[\,]_+$ est défini :

$$[x]_+ = x \quad si \quad x \geq 0$$
$$[x]_+ = 0 \quad si \quad x < 0.$$

La contrainte de l'équation 3.21 devient alors :

$$y_i((z_i.w) + b) \geq 1 - \xi_i, \; i = 1, \ldots, l. \tag{3.31}$$

Dans ce cas, la fonction objective à optimiser s'écrit :

$$L(w, \xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} \xi_i; \tag{3.32}$$

où C représente le facteur de Lagrangien associé au terme de pénalisation rajouté.

Cortes et Vapnik démontrent [27] que la fonction de décision dans ce cas est caractérisée par :

$$w = \sum_{i=1}^{l} \alpha_i y_i z_i, \tag{3.33}$$

et que le Lagrangien dual à maximiser s'écrit :

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j}^{l} \alpha_i \alpha_j y_i y_j (z_i.z_j) \tag{3.34}$$

avec les contraintes :

$$0 \leq \alpha_i \leq C,\ i = 1, \ldots, l, \tag{3.35}$$

et

$$\sum_{i=1}^{l} \alpha_i y_i = 0. \tag{3.36}$$

3.5.4 Le SVM non-linéaire

Boser at al. [18], ont su produire des frontières de décision non-linéaire avec le SVM (Figure 5). L'idée est d'utiliser un noyau de Mercer qui permet de projeter les données dans un espace éventuellement plus grand dans lequel une séparation linéaire des classes est possible [28]. Il est alors important que la formulation du produit scalaire dans l'équation 3.27 reste intacte après l'introduction du noyau. Boser et al. démontrent qu'un noyau de Mercer ayant la forme :

$$\phi(x).\phi(y) = k(x, x_i) \tag{3.37}$$

ne change pas la nature de la fonction objective à optimiser qui équivaut toujours à un problème de maximisation quadratique.

Le Lagrangien dual de la fonction objective à maximiser est alors :

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j}^{l} \alpha_i \alpha_j y_i y_j k(x_i, x_j). \tag{3.38}$$

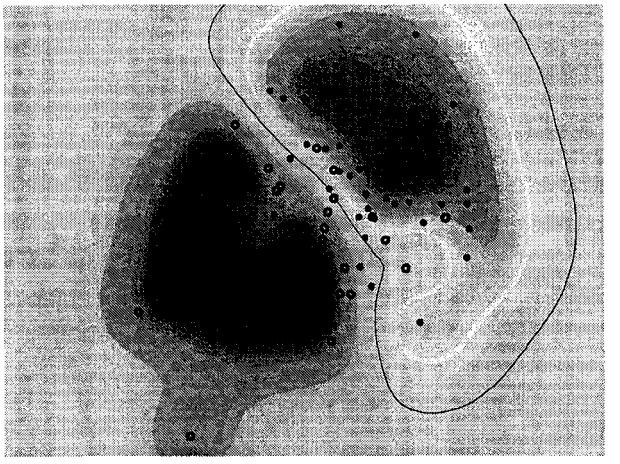


Figure 5 Frontière de décision non linéaire

La fonction de décision s'écrit encore :

$$f(x) = sgn\left(\sum_{i=1}^{l} y_i \alpha_i k(x, x_i) + b\right). \tag{3.39}$$

Notons que $f(x)$ est une combinaison linéaire de termes non-linéaires $k(x, x_i)$ qui représente la similarité entre les images des points x et x_i. Pour une observation de test x quelconque, la valeur de $f(x)$ indique la classe d'appartenance inférée par le SVM.

3.5.4.1 SVM L2

Pour des données non séparables, la variante L2 du SVM minimise :

$$L(w, \xi) = \frac{1}{C}\frac{\|w\|^2}{2} + \frac{1}{2}\sum_i \xi_i^2, \tag{3.40}$$

qui équivaut à la maximisation de la fonction objective :

$$W(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \widetilde{k}(x_i, x_j), \tag{3.41}$$

lorsque :

$$\widetilde{k}(x_i, x_j) = k(x_i, x_j) + \frac{1}{C} \quad if \quad i = j$$
$$\widetilde{k}(x_i, x_j) = k(x_i, x_j) \quad if \quad i \neq j.$$

avec les contraintes

$$\sum_i \alpha_i y_i = 0,$$

et

$$\alpha_i \geq 0, \forall i = 1, \ldots, l.$$

3.5.5 Conditions de Karush-Kuhn-Tucker

La résolution de l'optimisation quadratique de l'équation 3.38 est basée sur les conditions de convergence dites de «Karush-Kuhn-Tucker» (KKT) qui établissent les conditions nécessaires (mais parfois suffisantes) de convergence de la fonction objective duale. Ces conditions sont relativement simples et s'écrivent :

$$\begin{aligned} \alpha_i = 0 &\Rightarrow y_i f(x_i) \geq 1 \quad et\ \xi_i = 0 \\ 0 < \alpha_i < C &\Rightarrow y_i f(x_i) = 1 \quad et\ \xi_i = 0 \\ \alpha_i = C &\Rightarrow y_i f(x_i) \leq 1 \quad et\ \xi_i \geq 0 \end{aligned} \tag{3.42}$$

Les équations 3.42 reflètent une propriété importante du SVM stipulant qu'une grande proportion des exemples d'apprentissage sont situés en dehors de la marge et ne sont pas retenu par le modèle. Par conséquent, leurs multiplicateurs α_i sont nuls.

Les conditions de KKT traduisent le fait que seulement les variables α_i des points situés sur la frontière de la marge ($0 < \alpha_i < C$) ou à l'intérieure de celle-ci ($\alpha_i = C$) sont non nulles. Ces points sont les vecteurs de support du classifieur.

Le SVM produit alors une solution clairsemée n'utilisant qu'un sous ensemble réduit des données d'apprentissage. Sans cette propriété, l'entraînement du SVM sur de gros ensembles de données ainsi que son stockage deviennent extrêmement prohibitifs.

3.5.6 Calcul du biais b

Le paramètre de biais b permet de définir des surfaces de séparation ne passant pas par l'origine. Son calcul exploite les vecteurs de support respectant l'inégalité $0 \leq \alpha_i < C$ dont les ξ_i correspondants sont nuls. L'égalité suivante :

$$y_i(b + \sum_{j=1}^{l} y_j \alpha_j k(x_i, x_j)) = 1,$$

est alors vérifiée.

En considérant la moyenne calculée sur cet ensemble des vecteurs de support, une valeur stable de b peut s'écrire :

$$b = \frac{1}{\sharp sv}(y_i - \sum_{j=1}^{l} y_j \alpha_j k(x_i, x_j)),$$

où $\sharp sv$ représente le nombre de vecteurs de support considérés.

3.5.7 Le ν-SVM

Plusieurs modifications ont été proposées pour la formulation de base du SVM. Parmi celles-ci, le ν-SVM initialement prévu pour la régression mérite de s'y attarder [93].

L'idée derrière est d'introduire un hyper-paramètre qui permet de contrôler à la fois le nombre d'erreurs d'apprentissage et le nombre de vecteurs de support. Dans ce cas, la forme primaire de la fonction objective à minimiser s'écrit :

$$L(w, \xi, \nu, \rho) = \frac{1}{2}\|w\|^2 - \nu\rho + \frac{1}{l}\sum_{i=1}^{l}\xi_i, \tag{3.43}$$

avec les contraintes :

$$y_i(w.\phi(x_i) + b) \geq \rho - \xi_i$$
$$\xi_i \geq 0, i = 1, \ldots, l, \rho \geq 0.$$

Les auteurs démontrent que ν ($0 \leq \nu \leq 1$) est une borne supérieure de l'erreur d'apprentissage et une borne inférieure du nombre de vecteurs de support. Cette propriété permet de contrôler à la fois l'erreur d'apprentissage et la complexité du classifieur à travers le nombre de vecteurs de support.

La forme duale de la fonction objective à maximiser s'écrit alors :

$$\begin{aligned} & -\tfrac{1}{2}\textstyle\sum_{i,j}^{l} \alpha_i\alpha_j y_i y_j k(x_i.x_j) \\ \texttt{sujet à} \quad & 0 \leq \alpha_i \leq \tfrac{1}{n} \qquad i = 1, \ldots, l, \\ & \textstyle\sum_{i=1}^{l} \alpha_i y_i = 0, \\ & \textstyle\sum_{i=1}^{l} \alpha_i \geq \nu. \end{aligned}$$

3.6 Discriminant de Fisher non-linéaire (KFD)

L'idée du discriminant de Fisher non-linéaire (de l'anglais «Kernel Fisher Discriminant») est de résoudre le discriminant de Fisher classique dans l'espace augmenté F induit avec un noyau de Mercer quelconque [32]. La discrimination devient non-linéaire dans l'espace d'origine [68, 84, 12].

Le discriminant de Fisher désigné par LDA (de l'anglais «Linear Discriminant Analysis») vise a trouver l'hyper-plan qui maximise la variance inter-classe et minimise la variance intra-classe des données (Figure 6). Ceci est traduit par le critère de Rayleigh s'écrivant :

$$J(w) = \frac{w^T S_B w}{w^T S_W w}, \tag{3.44}$$

où

$$S_B = (m_2 - m_1)(m_2 - m_1)^T,$$

représente la variance inter-classe et

$$S_W = \sum_{k=1,2} \sum_{i \in I_k} (x_i - m_k)(x_i - m_k)^T,$$

représente la variance intra-classe. Ici, m_k et I_k sont respectivement la moyenne et l'ensemble des exemples de la classe k. Si les classes suivent deux distributions gaussiennes, le discriminant de Fisher est optimal au sens de Bayes. Le même constat peut être généralisé au cas multiclasse.

De la même façon que pour le SVM, il est possible d'appliquer l'algorithme du LDA dans l'espace augmenté induit par un noyau de Mercer quelconque en utilisant une expansion de la forme :

$$w = \sum_{i=1}^{l} \alpha_i \phi(x_i), \tag{3.45}$$

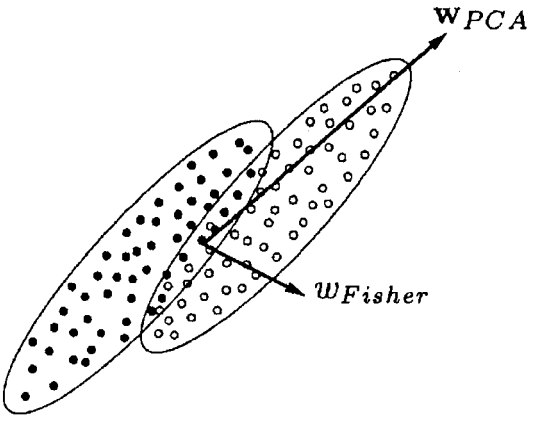


Figure 6 Discriminant de Fisher : W_{Fisher} et W_{PCA} sont respectivement le vecteur directionnel du discriminant de Fisher et l'axe principal du nuage de points

l étant le nombre total d'exemples.

En substituant l'équation 3.45 dans l'équation 3.44, nous trouvons le critère d'optimisation à maximiser suivant :

$$J(\alpha) = \frac{(\alpha^T \mu)^2}{\alpha^T N \alpha} = \frac{\alpha^T M \alpha}{\alpha^T N \alpha} \tag{3.46}$$

où

$$\mu_k = \frac{1}{|I_k|} K.1_k,\ N = K.K^T - \sum_{k=1,2} |I_k| \mu_k.\mu_k^T,\ \mu = \mu_1 - \mu_2,\ M = \mu.\mu^T.$$

K est la matrice dont les composantes K_{ij} sont égales à $\phi(x_i).\phi(x_j) = k(x_i, x_j)$.

$|I_k|$ représente la taille de l'ensemble de données appartenant à la classe k.

La projection d'une donnée de test sur l'hyper-plan de séparation donne :

$$w.\phi(x) = \sum_{i=1}^{l} \alpha_i k(x_i, x).$$

Dans le but de solutionner l'équation 3.46, on résout le système propre donné par

$$M\alpha = \lambda N\alpha$$

dont la solution est $\alpha = N^{-1}(\mu_2 - \mu_1)$. Toutefois, pour un grand ensemble de données, la solution est prohibitive en mémoire et en temps de calcul. Mika et al. [67, 66] ont pu transformer la fonction objective $J(\alpha)$ en un problème de programmation linéaire convexe rendant possible sa résolution.

3.7 Analyse en Composantes Principales non-linéaire

L'Analyse en Composantes Principales (ACP) est une technique descriptive qui calcule les axes d'inertie d'un nuage de points (l'axe W_{PCA} dans la figure 6 représente l'axe principal des points). Les axes d'inertie représentent des directions orthogonales suivant lesquelles la variance des données est maximale. Le terme «composante principale» est l'association d'une direction principale et de son amplitude. Les composantes principales est une solution au sens des moindres carrés qui revient à procéder à une rotation de l'ensemble des données suivie d'une translation. L'ACP peut être utilisée pour réduire la dimensionalité des données en conservant uniquement un sous-ensemble des composantes principales. Toutefois, l'ACP est une méthode linéaire qui ne détecte pas les structures non-linéaires des données.

L'ACP non linéaire rend possible l'extraction de ces propriétés grâce à l'utilisation de noyaux de Mercer de la même façon que dans le SVM ou le KFD.

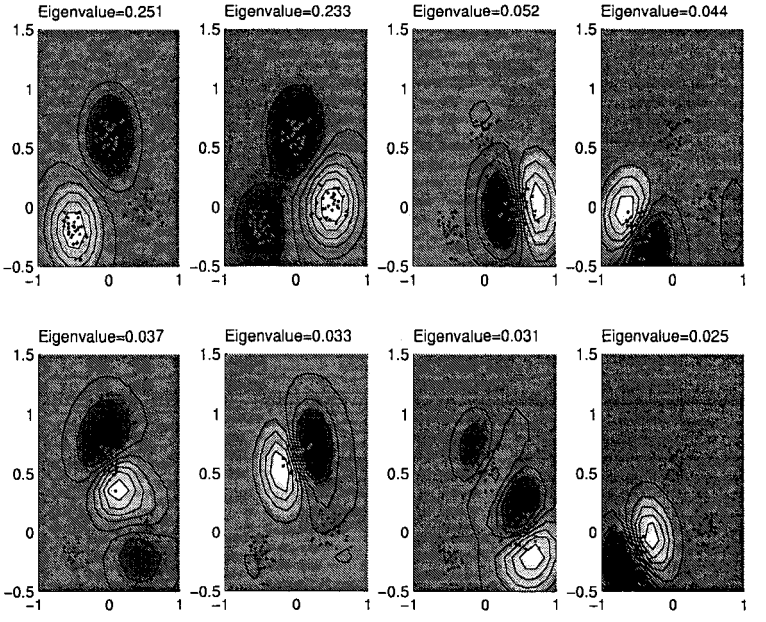


Figure 7 Découverte de structures par projection des données sur les huit premières composantes principales du KPCA

Soient alors les données $x_1, \ldots, x_l \in \mathbb{R}^N$ projetées dans l'espace augmenté F dans lequel est estimé la matrice de covariance donnée par :

$$C = \frac{1}{l} \sum_{j=1}^{l} \phi(x_j)\phi(x_j)^T.$$

Les composantes principales sont calculées en résolvant le système propre donné par :

$$\texttt{Trouver } \lambda > 0, \mathbf{V} \neq 0 \texttt{ avec}$$
$$\lambda \mathbf{V} = C\mathbf{V} = \frac{1}{l} \sum_{j=1}^{l} (\phi(x_j).\mathbf{V})\phi(x_j). \tag{3.47}$$

Les vecteurs propres calculés correspondent aux axes principaux et constituent une combinaison linéaire des images des données dans l'espace augmenté tel que :

$$\mathbf{V} = \sum_{i=1}^{l} \alpha_i \phi(x_i). \tag{3.48}$$

Les coefficients α_i de l'équation sont trouvés en calculant les valeurs propres λ du système propre :

$$\lambda\alpha = K\alpha,$$

où $\alpha = (\alpha_1, \ldots, \alpha_l)^T$. Il est d'usage de centrer les données dans l'espace F avant toute autre procédure.

Outre la réduction de la dimensionalité, l'ACP non-linéaire peut être utilisée pour l'extraction de caractéristiques de la façon suivante : Étant donné un exemple de test x, sa projection sur chacune des composantes principales de l'espace F constitue un vecteur de caractéristiques dont la $k^{ème}$ coordonnée s'écrit :

$$\mathbf{V}^k.\phi(x) = \sum_{i=1}^{l} \alpha_i^k(\phi(x_i).\phi(x)) = \sum_{i=1}^{l} \alpha_i^k k(x_i, x).$$

Il est utile de signaler que cette version de l'ACP donne lieu a l composantes principales au lieu de N dans l'ACP classique (N étant la dimension de l'espace de d'entrée). Celles-ci deviennent difficiles à calculer dès que nous disposons d'une grande quantité de données.

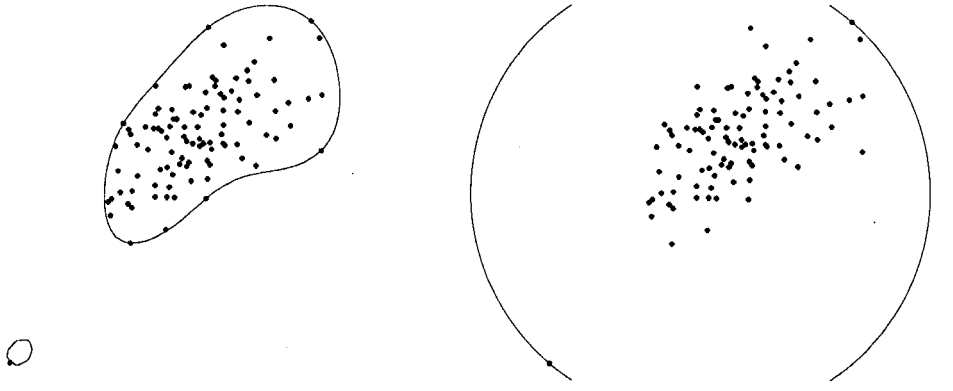
Smola et al. [99] avaient utilisé une approximation clairsemée de la matrice K qui conserve les principaux axes d'inertie. Aussi, Tipping et al. [112] avaient proposé une approche similaire inspirée de l'apprentissage Bayesien. Enfin, Smola et al. [97] avaient appliqué une régularisation de type $L1$ à la méthode dans le but de réduire la complexité de la solution.

3.8 Classification mono-classe

Les méthodes d'estimation de densité sont des techniques d'apprentissage non-supervisées. Cette estimation peut s'avérer problématique pour plusieurs raisons. D'abord, il n'existe pas toujours une densité (quelques distributions n'ont pas de densité de probabilité). Aussi, estimer avec précision cette densité n'est pas garantie. Par ailleurs, dans plusieurs applications, il est suffisant de calculer le support de la distribution de données. Le SVM mono-classe a été proposé pour la première fois par Vapnik pour l'estimation du contour d'une distribution de points [115]. Quelques années plus tard, deux autres algorithmes utilisant la théorie des noyaux ont été proposées.

3.8.1 Méthode de l'hyper-sphère

Cette méthode, proposée par Tax et al., isole les données dans une hyper-sphère de l'espace augmenté F [109]. La frontière obtenue dans l'espace d'entrée est non-linéaire et dépend du noyau utilisé (Figure 8).



(a) frontière induite par le noyau RBF (b) frontière induite par le noyau linéaire

Figure 8 Surface de l'hyper-sphère englobante dans l'espace d'entrée pour deux types de noyaux

La plus petite hyper-sphère dans F englobant les données est identifiée par son centre et son rayon. Son rayon est calculé en maximisant la fonction objective :

$$\sum_{i=1}^{l} \beta_i k(x_i, x_i) - \sum_{i,j=1}^{l} \beta_i \beta_j k(x_i, x_j)$$

sous les contraintes :

$$\sum_{i=1}^{l} \beta_i = 1 \forall i \; \beta_i \geq 0.$$

Le centre de l'hyper-sphère dans F est égal à :

$$\sum_{i=1}^{l} \beta_i \phi(x_i).$$

Les observations de l'ensemble des données pour lesquelles la variable β_i est différente de zéro sont les vecteurs de support de l'hyper-sphère. L'équation de l'hyper-sphère dépend de ce sous-ensemble de données. Ces points sont situés sur la surface de l'hyper-sphère et par conséquent sur la frontière de séparation qui isole les données (Figure 8).

Une version régularisée de l'algorithme permet d'isoler les données à l'intérieure de l'hyper-sphère tout en permettant quelques exemples d'être en dehors de celle-ci. Cependant, un facteur γ est introduit pour pénaliser l'éloignement de ces points de la surface de la boule. Dans ce cas, il existe une contrainte supplémentaire $\beta_i \leq \gamma$ [110].

3.8.2 Méthode de l'hyper-plan

Cette méthode a été proposée par Scholkopf et al. et vise à isoler les images des points dans F par un hyper-plan au lieu d'une hyper-sphère. Moyennant un noyau non linéaire, cet hyper-plan donne lieu à des contours fermés dans l'espace d'entrée.

Le calcul de l'hyper-plan se fait en minimisant la fonction objective quadratique s'écrivant :

$$\frac{1}{2}\|w\|^2 + \frac{1}{\nu l}\sum_i \xi_i - \rho \tag{3.49}$$

$$\texttt{sujet à} \quad w.\phi(x_i) \geq \rho - \xi_i, \xi_i \geq 0. \tag{3.50}$$

L'équation de l'hyper-plan dans l'espace d'origine s'écrit :

$$f(x) = \sum_i \alpha_i k(x_i, x) - \rho.$$

Les coefficients α_i sont trouvés en minimisant la fonction objective duale s'écrivant :

$$\frac{1}{2}\sum_{i,j} \alpha_i \alpha_j k(x_i, x_j) \tag{3.51}$$

$$\texttt{sujet à} \quad 0 \leq \alpha_i \leq \frac{1}{\nu l}, i = 1, \ldots, l$$

$$\sum_{i=1}^{l} \alpha_i = 1.$$

La variable ν désigne la proportion désirée des données situées au delà de l'hyper-plan et ρ est un scalaire proportionnel à la distance séparant l'hyper-plan de l'origine.

3.9 Algorithmes d'apprentissage du SVM

Afin de trouver les paramètres du SVM, il est nécessaire de résoudre le problème d'optimisation quadratique convexe donné par l'équation 3.38 dont la formulation matricielle

s'écrit encore :

$$-\frac{1}{2}\alpha^T \hat{K}\alpha + \mathbf{1}^{\mathbf{T}}\alpha,$$

où $\hat{K}$ est une matrice semi-définie positive dont les composantes $\hat{K_{ij}} = y_i y_j k(x_i, x_j)$ et $\mathbf{1}$ est le vecteur unité de même taille que α. Comme la fonction objective est convexe, tout maximum local est aussi un maximum global. Toutefois, il peut y avoir des solutions optimales différentes en terme de α_i donnant lieu à des performances différentes.

Il existe une grande variété de méthodes et de logiciels traitant de la résolution de problèmes quadratiques [1]. Cependant, quelques méthodes seulement sont capables de traiter un grand nombre d'exemples- souvent sous hypothèse que la matrice de Grahm Schmidt K soit creuse [114, 14]. Dans le cas contraire, l'apprentissage d'un SVM de quelques centaines d'exemples prendrait énormément de temps de calcul et assez de ressources mémoire. Seulement, il est possible de dériver des algorithmes qui exploitent la forme particulière de la fonction objective duale du SVM. Dans cette section, nous allons présenter trois approches différentes pour la résolution du problème quadratique du SVM. Ensuite, une attention particulière sera portée à l'algorithme de Joachims que nous utilisons pour l'entraînement du classifieur dans la partie expérimentale.

3.9.1 Méthode de chunking

La résolution de la fonction objective duale de l'équation 3.27 avec un très grand nombre d'exemples donne lieu à un vecteur α creux. Selon les données, plusieurs des paramètres α_i sont soit nuls ou égales à C. S'il y a moyen de savoir a priori lesquels α_i seront nuls, il est possible de réduire la taille de la matrice $\hat{K}$ sans altérer la valeur de la fonction objective. Aussi, une solution α est valide si et seulement si elle respecte les conditions de KKT. Vapnik [120] était le premier à décrire une méthode qui exploite cette propriété en prenant en compte seulement les α_i non nuls ou ceux violant les conditions de Karush Kuhn Tucker. La taille de ce sous ensemble dépend du nombre de vecteurs de support,

[1] Parfois on parle de programmation quadratique qui est généralement désignée QP

de la taille des données et de la complexité du problème de classification. Cette méthode se comporte assez bien sur des problèmes de quelques centaines de vecteurs de support. Des tâches plus complexes requièrent un schéma de décomposition de l'objective en sous problèmes plus facile à résoudre. Cette technique est décrite ci-dessous.

3.9.2 Méthode de décomposition successive

Cette méthode est similaire à celle du «Chunking» dans la mesure où elle considère aussi une succession de sous problèmes quadratiques à résoudre. La différence est que la taille des sous problèmes retenus est fixe. Cette méthode est basée sur la constatation qu'une succession de sous-problèmes quadratiques ayant au moins un exemple qui ne vérifie pas les conditions de KKT converge toujours vers une solution optimale [73, 74]. Osuna et al. suggèrent de conserver la taille du sous-problème fixe et d'ajouter ou d'enlever un exemple à la fois. Ceci permet d'entraîner de gros ensembles de données. En pratique, cette stratégie peut converger lentement si diverses heuristiques ne sont pas prises en compte. En effet, il est possible d'adopter des stratégies sophistiquées afin d'inclure ou d'exclure quelques exemples de la fonction objective. Différentes stratégies de caching peuvent aussi accélérer la convergence même avec quelques milliers de vecteurs de support. L'algorithme de SVM_{light} est une implémentation de cette méthode [46]. Un package d'optimisation quadratique reste toutefois nécessaire. Nous décrirons en profondeur cet algorithme dans la section 3.10.

3.9.3 Méthode de minimisation séquentielle : SMO

La méthode d'optimisation par minimisation séquentielle (de l'anglais «Sequential Minimal Optimization») proposée par Platt peut être perçue comme le cas extrême des méthode de décomposition successive [77]. A chaque itération, elle résout un problème quadratique de taille égale à deux. La résolution de ce dernier est analytique et donc nul besoin de recourir à un module d'optimisation quadratique. Encore faut-il choisir le bon couple de variables (α_i, α_j) à optimiser durant chaque itération. Les heuristiques que l'auteur utilise

sont basées sur les conditions de KKT. Une autre variante de la méthode a été proposée par Keerthi et al. et introduit quelques autres heuristiques à l'algorithme [50]. Son implémentation est relativement simple et un pseudo-code de la méthode est aussi fourni [77]. Des versions de l'algorithme adaptés à la régression [96, 98] ou aux problèmes mono-classe sont aussi disponibles [91].

3.10 Algorithme de Joachims

L'algorithme de Joachims [46] est une implémentation de SVM_{light}. Il peut traiter un gros ensemble de données allant jusqu'à quelques centaines de milliers d'exemples. Dans cette section, nous décrivons l'algorithme.

Comme vu précédemment, la fonction objective duale à minimiser s'écrit :

$$W(\alpha) = -\alpha^T \mathbf{1} + \tfrac{1}{2}\alpha^T K \alpha \tag{3.52}$$

$$\texttt{sujet à :} \qquad \alpha^T y = 0 \tag{3.53}$$

$$0 \leq \alpha \leq C\mathbf{1} \tag{3.54}$$

La taille du problème à optimiser dépend du nombre d'exemples d'apprentissage l. Comme la taille de la matrice K est l^2, résoudre 10000 données d'apprentissage ou plus devient impossible à moins de garantir un espace mémoire suffisant de l'ordre de 10^8. Une alternative serait d'estimer la matrice K chaque fois qu'il est nécessaire.

Une autre approche possible à cette limitation est de décomposer le problème en une série de tâches moins complexes et plus faciles à résoudre. SVM_{light} est une implémentation modifiée de l'algorithme de décomposition successive de Osuna et al. [73]. Cette méthode intègre les idées suivantes :

- une stratégie de sélection d'un ensemble actif (sous ensemble des données dont les multiplicateurs α_i sont variables).
- décomposition successive du problème. Ceci exploite les deux propriétés suivantes :
  1. il existe beaucoup moins de vecteurs de support que d'exemples d'apprentissage,
  2. beaucoup de vecteurs de support ont leurs multiplicateurs α_i égal à C.
- accélération de la convergence en utilisant la technique du 'caching' et la mise à jour incrémentale des valeurs de α_i.

3.10.1 Stratégie de décomposition

L'idée s'inspire de la stratégie de décomposition utilisée par Osuna et al. [73] qui utilise une stratégie similaire à celle des ensembles actifs [34]. A chaque itération, les paramètres α_i sont partagés en deux catégories.

- un ensemble B de paramètres variables.
- un ensemble N de paramètres fixes.

Les paramètres de B sont mis à jour à chaque itération, alors que les paramètres de N sont temporairement fixés à leurs valeurs précédentes. L'ensemble B est aussi appelé ensemble actif. Il a une taille q largement inférieure à l. L'algorithme fonctionne comme suit :

- Tant que les conditions de KKT ne sont pas remplies :
  - Sélectionner q paramètres pour l'ensemble actif B. Les variables $l-q$ restantes sont fixées à leurs valeurs courantes.
  - Décomposer le problème initial en sous-problème QP et optimiser sur l'ensemble actif B.
- Terminer et retourner α.

Comme la matrice Hessienne K est garantie semi-défini positive, et toute les contraintes de l'objective sont linéaires, le problème est convexe, et les conditions de KKT sont alors des conditions nécessaires et suffisantes d'optimalité.

3.10.2 Sous-problème QP

Si les conditions d'optimalité ne sont pas satisfaites, l'algorithme décompose la fonction objective originale et résout une succession de sous-problèmes dérivés.

La décomposition de la fonction objective originale $W(\alpha)$ (équation 3.52) est garantie de converger vers une solution optimale si l'ensemble actif B respecte quelques conditions [73]. En particulier, l'ensemble des paramètres variables B est séparé de N. Aussi, nous supposons que α, y et K sont adéquatement arrangés tel que :

$$\alpha = \begin{vmatrix} \alpha_B \\ \alpha_N \end{vmatrix}, \; y = \begin{vmatrix} y_B \\ y_N \end{vmatrix}, \; K = \begin{vmatrix} K_{BB} & K_{BN} \\ K_{NB} & K_{NN} \end{vmatrix} \tag{3.55}$$

Le problème d'optimisation quadratique original peut s'écrire :

$$\min_{\alpha} W(\alpha) = -\alpha_B^T(1 - K_{BN}\alpha_N) + \tfrac{1}{2}\alpha_B^T K_{BB}\alpha_B + \tfrac{1}{2}\alpha_N^T K_{NN}\alpha_N - \alpha_N^T \mathbf{1} \tag{3.56}$$

$$\texttt{sujet à :} \quad \alpha_B^T y_B + \alpha_N^T y_N = 0 \tag{3.57}$$

$$0 \leq \alpha \leq C\mathbf{1} \tag{3.58}$$

Comme les paramètres dans N sont fixes, les termes $\frac{1}{2}\alpha_N^T K_{NN}\alpha_N$ et $-\alpha_N \mathbf{1}$ sont constants et peuvent être omis de la fonction objective. Il en résulte un sous problème beaucoup

moins complexe qui peut être résolu avec des packages standards d'optimisation quadratique.

Le choix de l'ensemble actif B contenant les paramètres variables est traité ci-dessous.

3.10.3 Sélection de l'ensemble actif

La stratégie de sélection de l'ensemble actif consiste à choisir les paramètres α_i qui garantissent la convergence la plus rapide vers le minimum de $W(\alpha)$. Cette technique repose sur la méthode de Zoutendjik qui utilise une approximation du premier ordre de la fonction désirée [129]. L'idée est de trouver la direction de la plus grande pente de descente d utilisant seulement q valeurs non nulles parmi les α_i. Ces q paramètres constituent l'ensemble actif B. Cette approche nécessite la résolution de la fonction objective suivante :

$$\min_{\alpha} V(\mathbf{d}) = g(\alpha^t)^T \mathbf{d} \tag{3.59}$$

$$\texttt{sujet à :} \quad \mathbf{y}^T \mathbf{d} = 0 \tag{3.60}$$

$$d_i \geq 0 \qquad \textit{pour } i : \alpha_i = 0 \tag{3.61}$$

$$d_i \leq 0 \qquad \textit{pour } i : \alpha_i = C \tag{3.62}$$

$$-1 \leq d \leq 1 \tag{3.63}$$

$$|\{d_i : d_i \neq 0\}| = q \tag{3.64}$$

La fonction objective de l'équation 3.59 cherche une direction de descente. Une direction de descente a un produit scalaire négatif avec le gradient $g(\alpha^T)$ au point courant $\alpha^{(t)}$. Les contraintes des équations 3.60, 3.61 et 3.62 permettent de projeter la direction de descente sur le plan défini par l'égalité de l'équation 3.53 et de forcer la contrainte de l'équation 3.54. La contrainte donnée dans l'équation 3.63 normalise la taille du vecteur pour que

Tableau V

Durées d'apprentissage et nombre de vecteurs de support sur le problème «Income Prediction»

Exemples	SVM_{light}	SMO	Chunking	♯SV
1605	7.8	15.8	34.8	691
2265	16.8	32.1	144.7	1007
3185	30.6	66.2	380.5	1293
4781	68.4	146.6	1137.2	1882
6414	120.6	258.8	2530.6	2475
11221	430.8	781.4	11910.6	4182
16101	906.0	1784.4	N\A	5894
22697	1845.6	4126.4	N\A	8263
32562	3850.2	7749.6	N\A	11572

le problème d'optimisation soit bien posé. Enfin, l'équation 3.64 garantie que **d** contient seulement q variables non nulles. Ces dernières seront incluses dans l'ensemble actif B.

Le tableau V compare les durées d'apprentissage de SVM_{light} avec les algorithmes de «SMO» et de «Chunking» pour plusieurs tailles des données sur le problème «Income Prediction» de UCI [16]. Aussi, sont donnés les nombres de vecteurs de support pour chaque expérimentation.

3.11 Classification de données multiclasses

Le SVM est un classifieur binaire qui ne traite habituellement que des données appartenant à deux classes. Cependant, il existe des versions plus élaborées prenant en compte plus de deux classes simultanément au sein de la même fonction objective. Il n'est pas prouvé toutefois que celles-ci minimisent le risque structurel. La recherche demeure, cependant, très active sur ce sujet [44].

Nous cherchons à modéliser une fonction $G : \Omega \rightarrow 1, \ldots, K$, qui définit K partitions dans l'espace des caractéristiques Ω. Connaissant G, les partitions des classes sont définies par $G^{-1}(c)$, où $c \in [1..K]$.

Pour un problème à deux classes, l'hyper-plan (w, b) du SVM délimite les deux partitions selon $signf(x)$ où $f(x) = w.x+b$. Par ailleurs, bien que le SVM soit un classifieur binaire, il peut facilement être étendu pour décider de l'appartenance de données multiclasses. En particulier, on trouve deux schémas de classification :

1. *Un-contre-Tous.* Cette méthode est simple d'usage et donne des résultats raisonnables. Elle consiste à entraîner K SVM différents, séparant chaque classe des $K-1$ restantes. Ainsi, pour chaque exemple de test, K valeurs de sortie $f_c(x)$ sont disponibles. Une façon naïve mais simple de classification consiste à attribuer l'exemple à la sortie de plus grande amplitude.
2. *Un-contre-Un.* Cette méthode requiert l'apprentissage de $\frac{1}{2}K(K-1)$ classifieurs pour tous les couples de classes possibles. Durant le test, la méthode requiert la combinaison de toutes les sorties de classifieurs pour qu'une décision soit émise.

3.11.1 Approche Un-contre-Tous

L'idée de cette stratégie est de construire autant de classifieurs que de classes. Ainsi, durant l'apprentissage, tous les exemples appartenant à la classe considérée sont étiquetés positivement (+1) et tous les exemples n'appartenant pas à la classe sont étiquetés négativement (-1). A la fin de l'apprentissage, nous disposons de K modèles correspondant aux hyper-plans (W_i, b_i) tels que $i = 1, \ldots, K$.

Durant le test, l'exemple est associé à la classe dont la sortie est positive selon la règle

$x \in C_k$ si $W_i.x + b_i > 0$ pour $i = k$.

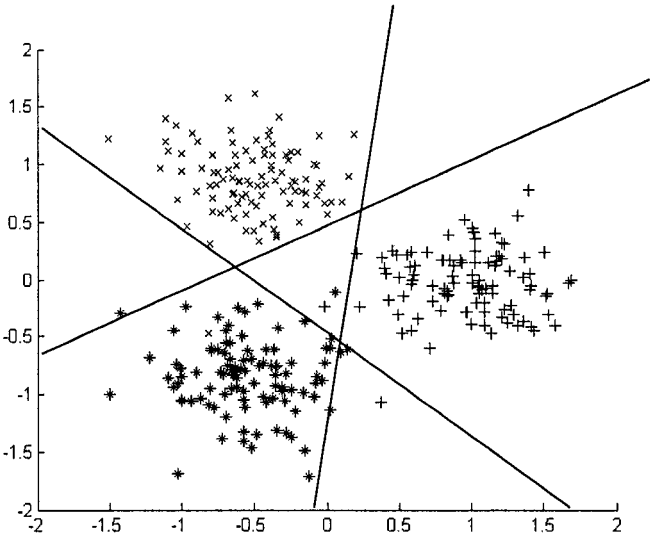


Figure 9 Problème à trois classes : frontières de décision linéaires dans la stratégie Un-contre-Tous

Or, il est possible que plusieurs sorties soient positives pour un exemple de test donné. Ceci est particulièrement le cas des données ambiguës situées près des frontières de séparation des classes. On utilise dans ce cas un vote majoritaire pour attribuer l'exemple x à la classe C_k selon la règle de décision

$$c = \arg\max_i (W_i.x + b_i).$$

Si dans beaucoup de cas, la règle énoncée çi-haut est suffisante, il se peut qu'elle faille lorsque les sorties des K classifieurs ne sont pas comparables. En effet, quelques chercheurs ont mis en évidence ce phénomène pour des données de difficulté moyenne ou grande. Il est facile d'expliquer ceci en considérant le SVM comme étant un perceptron opérant dans un espace défini par l'ensemble de vecteurs de support. D'ailleurs, une analogie intéressante peut être établie entre le SVM et le PMC dans la mesure où l'on sait que la couche de sortie du PMC est un perceptron opérant dans l'espace des neurones cachés. Les poids de sa couche de sortie sont alors équivalents aux paramètres α_i des vecteurs de support. Dans l'approche Un-contre-Tous, les différents SVM sont entraînés indépen-

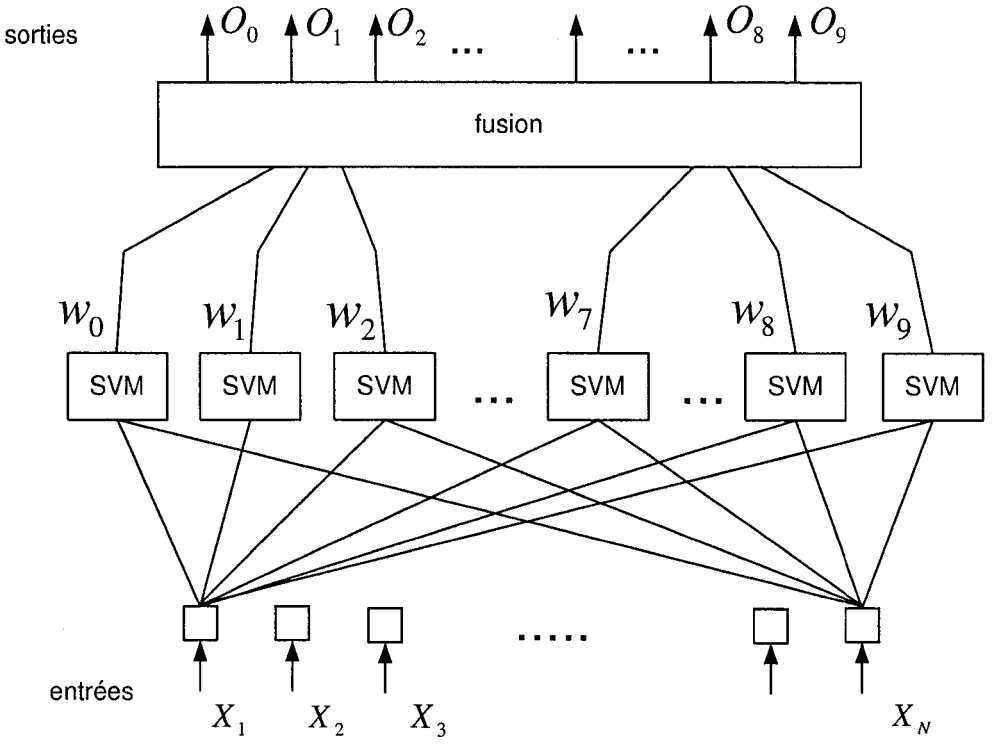


Figure 10 Architecture du système en stratégie Un-contre-Tous

damment les uns des autres. Ils produisent alors des ensembles de vecteurs de support différents. Il en résulte que leurs sorties fournissent des mesures dont les métriques ne sont pas comparables. Nous reportons plus bas les techniques utilisées pour remédier à cet inconvénient.

La figure 10 montre l'architecture du système en stratégie Un-Contre-Tous. Notons, que l'étage étiqueté *«fusion»* désigne le schéma de vote utilisé et aussi toute sorte d'expert capable de fusionner les sorties pour décider de la classe d'appartenance.

3.11.1.1 Normalisation par la marge

Il est possible de transformer les sorties brutes en utilisant une heuristique qui consiste à normaliser les marges des classifieurs $\|w\|^2$ à 1. Ainsi, la sortie $f(x)$ peut être pondérée

par le facteur de normalisation donné par :

$$\frac{1}{\pi^2} = \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j),$$

où i et j sont les indices des exemples d'apprentissage. La sortie normalisée du SVM devient alors :

$$\tilde{f}(x) = \pi f(x).$$

3.11.1.2 Estimation de la probabilité à posteriori

Nous allons voir dans le chapitre 5 (section 5.3.2) qu'il est possible de convertir la sortie brute du SVM en probabilité en utilisant une sigmoïde de la forme :

$$g(x) = \frac{1}{1 + \exp(Ax + B)}.$$

Cette alternative permet de s'affranchir de la normalisation telle que expliquée çi-haut. Le paramètre A permet d'ajuster la pente de la sigmoïde à la marge du SVM. Notons aussi que ses paramètres sont appris sur un ensemble de validation ou sur un ensemble d'apprentissage assez grand.

3.11.2 Approche Un-contre-Un

L'approche Un-Contre-Un est un cas spécial des méthodes de décomposition proposées par Dietterich et al. [29] pour résoudre des problèmes à plusieurs classes. Cette approche requiert la construction de $\frac{1}{2}K(K-1)$ SVM chacun séparant un couple de classes (i, j) parmi ceux existants.

Pendant la classification, un vecteur d'entrée x est présenté à l'ensemble des classifieurs construits. La sortie de chaque SVM fournit un vote partiel concernant uniquement le couple de classes (w_i, w_j). En considérant que chaque SVM calcule un estimé $\hat{p}_{ij}$ de la

probabilité :

$$p_{ij} = P(x \in w_i | \mathbf{x}, \mathbf{x} \in w_i \cup w_j), \tag{3.65}$$

alors le règle de classification la plus simple peut s'écrire :

$$\arg \max_{1 \leq i \leq K} \sum_{j \neq i} [\hat{p}_{ij} > 0.5]. \tag{3.66}$$

L'opérateur [ ] est défini :

$$[\eta] = \begin{cases} 1 & si\ \eta\ est\ vrai \\ 0 & sinon \end{cases}$$

Cette combinaison considère que les sorties des SVM sont des valeurs binaires de 1 ou 0. Une autre approche de reconstruction pourrait tirer avantage de l'information de confiance associée à chacune des sorties $\hat{p}_{ij}$. Dans l'hypothèse que ces valeurs représentent des probabilités, il est possible d'estimer une approximation $\hat{p}_i$ de la probabilité à posteriori

$$p_i = P(x \in w_i | x).$$

En considérant la matrice carrée $\hat{\mathbf{P}}$ avec les entrées $\hat{p}_{ij}$ tels que $(i,j)_{i,j=1,\ldots,K, i \neq j}$ et avec $\hat{p}_{ji} = 1 - \hat{p}_{ij}$, les valeurs de $\hat{p}_i$ peuvent être calculées par :

$$\hat{p}_i = \frac{2}{K(K-1)} \sum_{j \neq i} \hat{p}_{ij}, \tag{3.67}$$

et la règle de décision s'écrit :

$$\arg \max_{1 \leq i \leq K} \hat{p}_i. \tag{3.68}$$

Nous appellerons cette procédure *«reconstruction souple»* par opposition à la *«reconstruction brute»* de l'équation 3.66. Les formulations des équations 6.2 et 6.3 peuvent être généralisées de la façon suivante : $\arg \max_{1 \leq i \leq K} \hat{p}_i$, $\hat{p}_i = \frac{2}{K(K-1)} \sum_{j \neq i} \sigma(\hat{p}_{ij})$, où σ est une fonction de seuillage binaire centrée sur 0.5 pour une reconstruction brute, ou bien une fonction identité pour la reconstruction souple.

Ces deux schémas de reconstruction exploitent différemment l'information disponible. Hasti et al. dans [41] démontrent que pour un problème à trois classes et une observation x de test dont la matrice **P** des probabilités $\hat{p}_{ij}$ est donné par :

$$\begin{pmatrix} - & 0.6 & 0.6 \\ 0.4 & - & 0.9 \\ 0.4 & 0.1 & - \end{pmatrix},$$

x est classifié dans w_1 avec une reconstruction brute, alors qu'une reconstruction souple donnera w_2.

Aussi, nous avons considéré trois fonctions de reconstruction supplémentaires σ telles que proposées dans [70] (tableau VI).

Les schémas $PWC4$ et $PWC5$ permettent de pondérer différemment les réponses positive et négative. La figure 11 présente l'architecture simplifiée du système en stratégie Un-contre-Un.

3.11.3 Calcul des probabilités à posteriori des classes

Dans l'approche Un-contre-Un, le SVM produit un score qui représente une mesure de distance de l'hyperplan. Sa valeur peut toutefois être convertie en mesure de probabilité grâce à l'utilisation de la procédure d'estimation de probabilité expliquée dans la section

Tableau VI

Fonctions d'activation pour la reconstruction

fonction	expression
PWC1	$\sigma(x) = \begin{cases} 1 & \text{si } x \geq 0.5 \\ 0 & sinon \end{cases}$
PWC2	$\sigma(x) = x$
PWC3	$\sigma(x) = \frac{1}{1+e^{-12(x-0.5)}}$
PWC4	$\sigma(x) = \begin{cases} 1 & \text{si } x \geq 0.5 \\ x & sinon \end{cases}$
PWC5	$\sigma(x) = \begin{cases} x & \text{si } x \geq 0.5 \\ 0 & sinon \end{cases}$

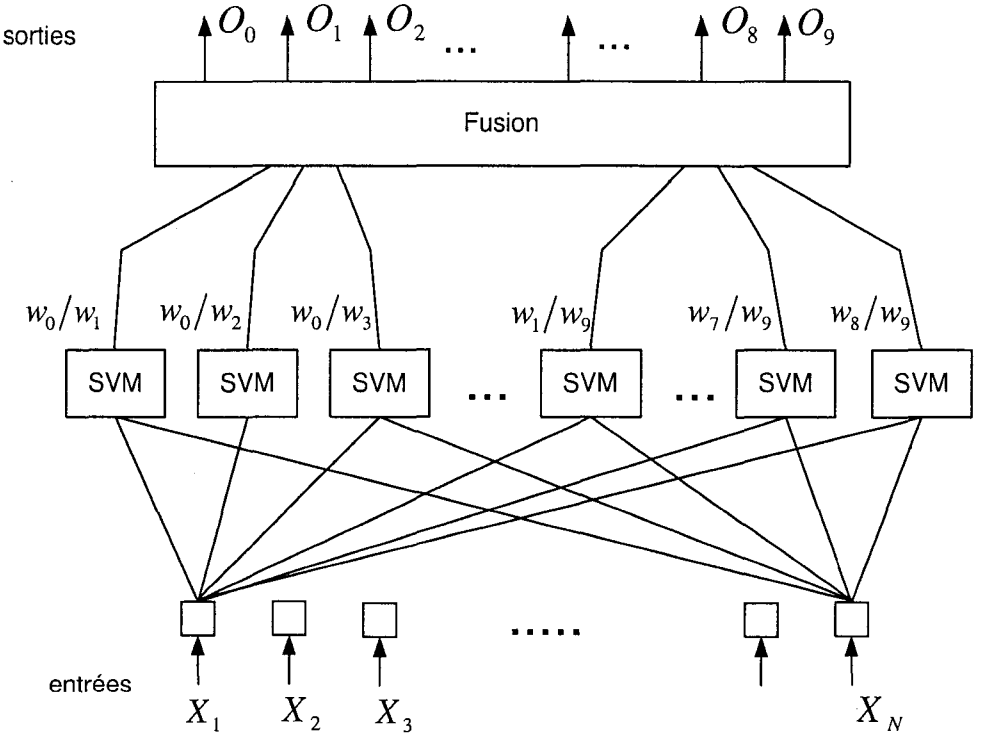


Figure 11 Architecture du système en stratégie Un-contre-Un

5.3.2. Une façon simple serait d'estimer la probabilité à posteriori comme :

$$\hat{p}_i = \frac{2}{K(K-1)} \sum_{j \neq i} \hat{p}_{ij}.$$

Cette formule pondère équitablement les estimations $\hat{p}_{ij}$ et ne néglige pas les mesures qui ne sont pas pertinentes. Ceci est particulièrement le cas lorsque x n'appartient à aucune des classes i et j. L'algorithme de Hasti et Tibshirani [41] a été proposé pour corriger cet inconvénient en utilisant une procédure itérative pour l'estimation des probabilités à posteriori des classes.

La probabilité que x appartienne à w_i sachant que x est de la classe w_i ou de la classe w_j, s'écrit :

$$\hat{p}_{ij} = P(w_i | x, x \in w_i \cup w_j), j \neq i.$$

Étant donné la matrice **P** des probabilités $\hat{p}_{ij}$, nous décrivons la procédure utilisée pour estimer les probabilités à posteriori P_i.

Soit les variables auxiliaires :

$$\mu_{ij} = \frac{P_i}{P_i + P_j},$$

pour trouver les P_i, il s'agit d'estimer les μ_{ij} correspondants qui sont le plus proche des valeurs de $\hat{p}_{ij}$. Ceci est traduit par la distance de Kullback-Lebleir entre les estimées μ_{ij} et $\hat{p}_{ij}$ donnée par :

$$E_{KL} = \sum_{i<j} n_{ij} [\hat{p}_{ij} \log \frac{\hat{p}_{ij}}{\mu_{ij}} + (1 - \hat{p}_{ij}) \log \frac{1 - \hat{p}_{ij}}{1 - \mu_{ij}}]. \tag{3.69}$$

Les valeurs de P_i peuvent être estimées en minimisant l'équation 3.69 selon la procédure décrite dans l'algorithme ci-dessous.

Algorithme

1) Initialiser $\hat{P}_i$ pour $i = 1, \ldots, K$, et calculer $\hat{\mu}_{ij}$ correspondants,

2) répéter jusqu'à convergence :

a) $\hat{P}_i \leftarrow \hat{P}_i . \frac{\sum_{j \neq i} n_{ij} \hat{p}_{ij}}{\sum_{j \neq i} n_{ij} \hat{\mu}_{ij}}$,

b) normaliser $\hat{P}_i$,

c) mise à jour de $\hat{\mu}_{ij}$.

3) pour $i = 1, \ldots, K$, faire $\hat{P}_i \leftarrow \frac{\hat{P}_i}{\sum_i \hat{P}_i}$.

Pour des raisons de simplicité, nous considérons $n_{ij} = 1 \ \forall i, j$. Aussi, nous utiliserons l'équation 6.2 pour estimer les variables $\hat{\mu}_{ij}$ de départ. Une fois que les probabilités à posteriori $\hat{P}_i(x)$ sont estimées pour tout $i = 1, \ldots, K$, la règle de classification suivante peut être appliquée :

$$c = \arg\max_i [\hat{P}_i(x)]. \tag{3.70}$$

3.11.4 Couplage optimal des classes

Dans l'algorithme de Hastie ci-haut, les coefficients n_{ij} de l'équation 3.69 peuvent s'écrire :

$$N = \begin{bmatrix} - & n_{01} & n_{02} & \cdots & n_{09} \\ n_{10} & - & n_{12} & \cdots & n_{19} \\ n_{20} & n_{21} & - & \cdots & n_{29} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ n_{90} & n_{91} & n_{92} & \cdots & - \end{bmatrix} \tag{3.71}$$

et reflètent l'importance de chaque classifieur (i, j) sur la décision finale étant donné un exemple de test x. Habituellement, ces coefficients sont considérés égaux et sont tous initialisés à un $\forall i, j$, tel que $i \neq j$. Donc, tous les classifieurs ont des contributions égales. Par ailleurs, étant donné une donnée de test $x \notin w_i \cup w_j$, l'estimé $\hat{p}_{ij}$ n'est pas pertinent et

ne reflète pas une information utile parce que le classifieur (i, j) n'est pas appris sur des exemples de classe autre que i et j. Par conséquent, les utiliser pour estimer les probabilités à posteriori peut biaiser le calcul. En réalité, étant donné une observation de test, pas toutes les sorties $\hat{p}_{ij}$ des SVM sont informatives, et pertinentes. Certaines d'entre elles peuvent biaiser le résultat surtout pour les exemples ambigus ou difficiles.

Li et al. dans [57] présentent un algorithme de couplage qui prend en compte ce fait. L'idée est de construire des classifieurs dédiés qui n'utilisent que les valeurs $\hat{p}_{ij}$ pertinentes à la classe qu'on cherche à calculer. En effet, soit la matrice $\mathbf{P}$ des probabilités $\hat{p}_{ij}$ connue. Étant donnée la matrice N des coefficients de pondération dans l'équation 3.71, en utilisant l'algorithme de couplage de Hasti et Tibshirani, nous sommes en mesure de construire un classifieur unique. Nous reprenons la définition de [57] sur le classifieur optimal :

Définition 1 *La matrice optimale N^i de coefficients n_{kl} pour la classe w_i s'écrivant*

$$N^i = \{n_{kl}\},$$

vérifie :

$$\begin{cases} n_{kl} = 1, & \textit{si k=i ou l=i}, \\ n_{kl} = 0, & \textit{sinon}, \end{cases}$$

tel que $k, l = 1, \ldots, K$ et $k \neq l$.

Le classifieur correspondant est appelé classifieur optimal et sera désigné O-PWC (pour «Optimal Pairwise Classification»)dans ce qui suit.

Ainsi, en considérant un problème à cinq classes, les matrices optimales de coefficients pour les classes 2 et 4 sont respectivement :

$$\begin{bmatrix} - & 1 & 0 & 0 & 0 \\ 1 & - & 1 & 1 & 1 \\ 0 & 1 & - & 0 & 0 \\ 0 & 1 & 0 & - & 0 \\ 0 & 1 & 0 & 0 & - \end{bmatrix}, \begin{bmatrix} - & 0 & 0 & 1 & 0 \\ 0 & - & 0 & 1 & 0 \\ 0 & 0 & - & 1 & 0 \\ 1 & 1 & 1 & - & 1 \\ 0 & 0 & 0 & 1 & - \end{bmatrix}.$$

Donc, étant donné un exemple de test $x \in w_i$, nous prendrons en compte uniquement les sorties des SVM dont l'une des classes d'apprentissages est w_i. Le restant des classifieurs est tout simplement ignoré. Pour chaque classifieur optimal, $O-PWC^i$, nous considérons seulement les probabilités $\hat{p}_{ij}$ ou $\hat{p}_{ji}$, tel que $j \neq i$ et $j = 1, \ldots, K$.

La sortie de $O - PWC^i$ est un vecteur de probabilités $(P_1^i, P_2^i, \ldots, P_K^i)$ où $i = 1, \ldots, K$, où chaque composante représente la probabilité d'appartenance à la classe respective sachant le modèle de classe w_i. Nous aurons en sortie dix vecteurs de probabilité correspondant aux classes.

3.11.4.1 Règle de décision

Si l'exemple de test appartient à la classe w_1, son vrai vecteur de probabilités est $T = (1, 0, .., 0)$, alors que pour un exemple de la classe w_2 T égal à $(0, 1, 0, 0, .., 0)$. Durant le test, nous utiliserons la règle de décision :

$$w_j = \arg\min_i [D_i]$$

où

$$D_i = \sum_k d_k(T, P^k).$$

Dans l'expression précédente, d_k représente une mesure de distance entre le vecteur de probabilités désiré et la sortie du $k^{ème}$ classifieur optimal. Comme mesure de distance, il est possible d'adopter des métriques différentes telles que l'erreur quadratique donnée par :

$$d_k = \sum_{i=1}^{K} (P_i^k(x) - T_i(x))^2,$$

ou bien l'entropie croisée qui s'écrit :

$$d_k = -\sum_{i=1}^{K} T_i(x) \log \frac{P_i^k}{T_i(x)}.$$

La figure 12 montre l'architecture du système avec cette stratégie pour un problème à dix classes (0 à 9).

3.12 Conclusion

Ce chapitre est organisé selon trois parties majeures. Après un bref historique sur le SVM, la première partie introduit la théorie de l'apprentissage statistique et le principe du risque structurel. La notion de régularisation par la maximisation de la marge y est expliquée. Ensuite, nous analysons le principe de la transformation de données dans l'espace augmenté à l'aide des noyaux de Mercer. Les propriétés de KMOD, notre nouveau noyau de classification, y sont étudiées dans la section 3.4. Plus loin, la formulation mathématique du SVM y est présentée. Aussi, nous y trouvons une brève discussion du ν-SVM, une variante qui permet de contrôler le nombre de vecteurs de support et l'erreur d'apprentissage simultanément. En outre, les versions non linéaires de l'ACP (KPCA) et du LDA (KFD) sont brièvement expliquées.

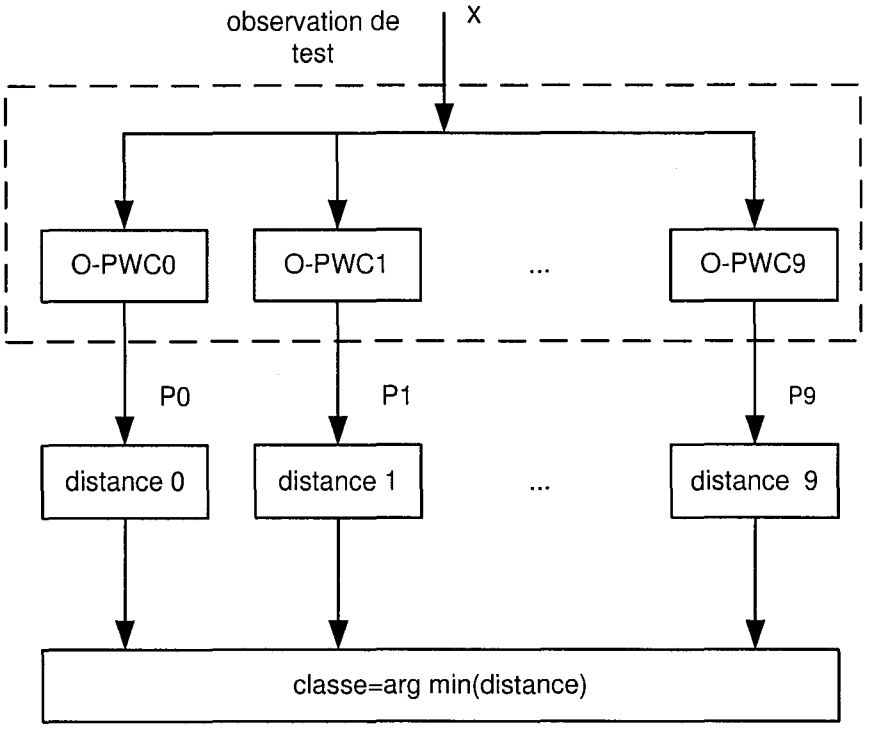


Figure 12 Architecture du système lors du couplage optimal de classes

Dans la deuxième partie nous discutons des stratégies d'apprentissage du SVM sur de gros ensembles de données. En particulier, nous détaillons l'algorithme de Joachims utilisé dans notre système.

Enfin, dans la dernière partie, nous abordons les différentes stratégies de combinaison pour la classification de données multiclasses. En particulier, nous expliquons l'approche un-contre-un adoptée dans notre travail et discutons les différents schémas de vote afin de fournir une classe d'appartenance.

CHAPITRE 4

SÉLECTION DE MODÈLES POUR LES MACHINES À VECTEURS DE SUPPORT : ÉTAT DE L'ART

4.1 Introduction

Parmi les difficultés majeures liées à l'utilisation de classifieurs figure la nécessité d'adapter les variables conditionnant le processus d'apprentissage et ne figurant pas dans la fonction de décision finale. Ces variables sont appelées hyper-paramètres. Si adéquatement choisies, elles permettent d'éviter les situations de sur-apprentissage fréquentes lorsque les données sont bruitées. Parmi les hyper-paramètres du SVM on trouve les paramètres de noyaux et la variable de compromis C.

Dans ce qui suit, nous présentons l'optimisation des valeurs des hyper-paramètres comme un problème de sélection de modèles. Ainsi, une revue de littérature approfondie des principales approches de sélection de modèles est fournie. Le développement du SVM étant très récent, la majorité des approches étudiées font référence au PMC. Dans un deuxième temps, nous abordons quelques uns des critères de sélection de modèles du SVM.

4.2 Sélection de modèles

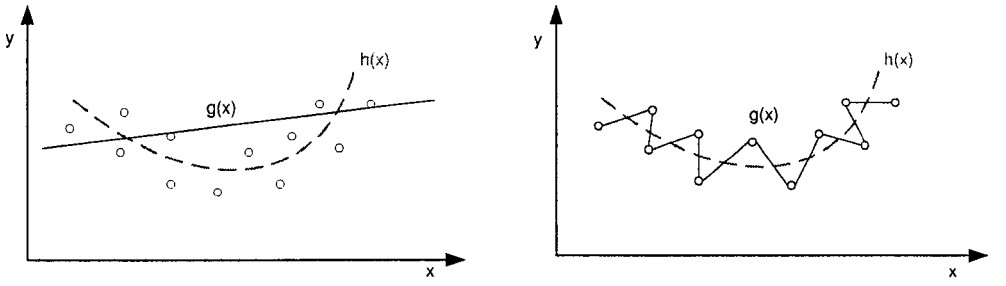
La problématique de sélection de modèles a connu un intérêt particulier avec l'adoption des réseaux de neurones comme modèles de classification et de régression. Dès lors, plusieurs chercheurs se sont intéressés aux moyens de choisir une architecture adéquate susceptible d'améliorer la généralisation du PMC. Plus particulièrement, le choix du nombre de couches cachées et le nombre de neurones cachés constituaient un vrai dilemme. Même si le théorème de Kolmogorov [51] soutient qu'un PMC d'une seule couche cachée est capable d'approximer des fonctions non-linéaires arbitraires, il demeure qu'il ne spécifie

rien sur le nombre de neurones cachées nécessaires. Les travaux de Vapnik ont permis de lier la généralisation des classifieurs à la notion de capacité [120].

Pour un PMC, cette dernière dépend du nombre d'exemples d'apprentissage, des paramètres de la sigmoïde des neurones cachés et du nombre d'époques d'apprentissage. Un choix non judicieux peut accentuer l'erreur. Une capacité exagérée cause un sur-apprentissage des données et la solution devient sensible au bruit. On parle alors d'erreur de variance. Une capacité non suffisante cause un sous-apprentissage des données. La solution, dans ce cas, approxime mal les exemples d'apprentissage. On parle alors d'erreur de biais. Une erreur de biais importante correspond à une erreur de variance réduite et vice versa. C'est le dilemme «Biais-Variance» bien connu en théorie d'apprentissage (Figure 13).

En pratique, il est nécessaire de chercher un compromis entre simplicité et complexité de la capacité des classifieurs. En régularisation, il est possible de considérer un terme de pénalisation dont l'amplitude sert à contrôler cette complexité par le biais d'un ou de plusieurs hyper-paramètres [79]. La technique du *«weight decay»* proposée par Hinton est probablement la première méthode de sélection de modèles à avoir été largement expérimentée [43]. La méthode a aussi été utilisée pour optimiser les poids d'un PMC en classification ou en régression linéaire en pénalisant la norme L2 des paramètres. Ce terme de régularisation peut aussi être interprété comme le logarithme de la probabilité a priori des paramètres du modèle [62]. Par ailleurs, la vraisemblance des données, connaissant le modèle, traduit le degré de fidélité aux données d'apprentissage.

Plusieurs critères de sélection de modèles sont utilisables pour le choix des valeurs des hyper-paramètres. Si le modèle utilise un seul hyper-paramètre, il est possible d'essayer un nombre fini de valeurs et choisir celle qui minimise le critère. Cette technique devient toutefois difficile à implémenter pour deux hyper-paramètres ou plus.



(a) Erreur de biais importante (b) Erreur de variance importante

Figure 13 Illustration du dilemme «Biais-Variance». $h(x)$ représente la vraie fonction des points observés (x, y) alors que $g(x)$ représente une approximation de $h(x)$ par interpolation des points observés

Afin de choisir les valeurs des hyper-paramètres, il est nécessaire de spécifier un critère permettant de sélectionner un modèle parmi plusieurs possibles. Dans le cas d'un SVM avec un noyau Gaussien par exemple, chaque valeur de σ correspond à une solution possible.

Idéalement, le risque réel décrit par :

$$R(\alpha, \theta) = \int \frac{1}{2} |f_{\alpha,\theta}(x) - y| dP(x, y).$$

est le meilleur critère qui puisse être utilisé (α et θ étant respectivement les paramètres et les hyper-paramètres du classifieur). Cependant, étant donné que la distribution $P(x, y)$ des données n'est pas connue, $R(\alpha, \theta)$ ne peut être estimée. Mais il existe, dans la littérature, plusieurs alternatives permettant de contourner cette limitation. Ainsi, il est possible d'utiliser une approximation analytique de l'erreur, une limite supérieure de l'erreur ou un estimé empirique de l'erreur. Dans ce qui suit, nous présentons un aperçu des principales techniques utilisées pour l'optimisation de classifieurs.

4.2.1 Approche par erreur de validation

Dans le but de trouver le SVM qui généralise le mieux, le plus simple serait d'utiliser un ensemble indépendant de celui de l'apprentissage pour l'évaluation du classifieur. Plusieurs SVM peuvent donc être entraînés sur un même ensemble d'apprentissage avec toutefois différentes valeurs d'hyper-paramètres. Leurs performances sont par la suite comparées en estimant l'erreur de chacun d'eux sur un ensemble de validation indépendant de celui de l'apprentissage. Le SVM qui fournit la plus faible erreur est le plus approprié à choisir. Cette approche est aussi connue sous le nom *«hold out method»*.

Afin d'éviter le sur-apprentissage des exemples de validation, le classifieur choisi est à son tour évalué sur un ensemble de test indépendant. Cependant en pratique, nous ne disposons pas toujours d'un nombre suffisant d'exemples étiquetés pour s'en réserver une partie pour la sélection de modèle. Dans ce cas, il est possible de diviser l'ensemble de données en K partitions distinctes pour en utiliser $K-1$ partitions pour l'apprentissage et la dernière pour évaluer la performance du classifieur. Ce processus est répété k fois pour chacun des K segments disponibles pour le test. L'erreur de test finale est la moyenne des K estimations possibles. Cette procédure permet d'utiliser une grande proportion des données pour l'apprentissage tout en utilisant l'ensemble des données pour l'estimation de l'erreur de validation croisée. Cette méthode requiert K procédures d'apprentissage. Elle est relativement lente pour un grand ensemble de données. Le cas limite où K est égal à la taille de l'ensemble de données $\mathcal{D}$ est appelé «LOO» (de l'anglais «leave one out»), et donne une estimation très proche de la vraie erreur.

Pour les K paires possibles (S_i^1, S_i^2) où $i = 1 \ldots K$ tel que $S_i^1 \bigcup S_i^2 = \mathcal{D}$, l'erreur de validation croisée s'écrit :

$$E_{cv}(\theta, D) = \frac{1}{K}\sum_{i=1}^{K}\frac{1}{|S_i^2|}\sum_{x_y \in S_i^2} Q(\alpha(\theta, S_i^1), x_y), \tag{4.1}$$

la variable α représente le vecteur des paramètres du classifieur, θ représente le vecteur des hyper-paramètres et x_y représente le couplet de donnée (x, y) avec x le vecteur d'entrée et y son étiquette désirée.

4.2.2 Approche algébrique

Il existe plusieurs estimateurs algébriques de l'erreur de généralisation. L'avantage majeur de cette catégorie d'estimateurs est la disponibilité de l'ensemble des données pour l'apprentissage. Ceci n'est pas la cas si un ensemble de validation est utilisé.

Parmi les critères algébriques, nous pouvons citer l'indicateur statistique C_p [64], l'erreur de prédiction finale $FPER$ (de l'anglais «final prediction error») [2], le critère d'information de Akaike [3], l'erreur prédite au carré PSE (de l'anglais «predicted squared error») [11], etc. Ces critères tentent d'estimer l'erreur de généralisation comme étant :

$$Estimé\ de\ l'erreur \quad = \quad Erreur\ d'apprentissage + terme\ de\ complexité,$$

où le terme de complexité reflète une pénalité qui croît avec le nombre de paramètres du modèle. Si le modèle est très simple, le critère donne une estimation très grande de l'erreur à cause de l'erreur d'apprentissage résiduelle. Dans le cas d'un modèle trop complexe, l'erreur est grande à cause du terme de complexité. Le minimum de l'erreur estimée est un compromis entre ces deux effets concourants. Pour une erreur quadratique, une formu-

lation typique du critère peut prendre la forme :

$$PE = \frac{2E}{N} + \frac{2W}{N}\sigma^2,$$

où E est l'erreur d'apprentissage, N est le nombre total d'exemples d'apprentissage, W est le nombre total de paramètres dans le modèle et σ^2 étant la variance du bruit des données.

Moody et al. [69] ont généralisé ce critère au PMC, qu'ils ont appelé GPE (de l'anglais «Generalized Prediction Error»). Il s'écrit :

$$GPE = \frac{2E}{N} + \frac{2\gamma}{N}\sigma^2,$$

où γ représente le nombre effectif de paramètres du PMC. Pour un PMC linéaire cette quantité est égale à :

$$\gamma = \sum_{i=1}^{W} \frac{\lambda_i}{\lambda_i + \mu}, \tag{4.2}$$

où λ_i représente les valeurs propres de la matrice Hessienne de l'erreur non régularisée et μ représente un paramètre de régularisation.

4.2.3 Approche bayesienne

La théorie bayesienne est due à Thomas Bayes (1702-1761), qui relayé par Pierre Simon Laplace (1749-1827) ont mis les fondements de la théorie d'inférence statistique. En l'an 1939, H. Jeffreys fit une étude approfondie de la théorie de la probabilité et mit en avant ce formalisme. En l'an 1992, MacKay propose une méthode d'apprentissage des poids du PMC basée sur la théorie de Bayes [63, 61]. La même année, il introduit une méthode inspirée de la même technique pour la comparaison de modèles [62]. Le même auteur remarqua que l'évidence pouvait être utilisée pour calculer l'amplitude des coefficients de régularisation dans la technique du «weight decay» et pour choisir un nombre adéquat

de neurones cachés dans le PMC. En 1997, il propose un algorithme utilisant le formalisme de l'évidence pour sélectionner les valeurs des hyper-paramètres dans une machine d'apprentissage.

Ainsi, selon la règle de Bayes, la probabilité a posteriori des paramètres w du modèle $\mathcal{H}_i$ s'écrit :

$$P(w|\mathcal{D}, \mathcal{H}_i) = \frac{P(\mathcal{D}|w, \mathcal{H}_i)P(w|\mathcal{H}_i)}{P(\mathcal{D}|\mathcal{H}_i)}, \tag{4.3}$$

traduisant la relation :

$$Probabilité\ a\ posteriori = \frac{Vraisemblance \times Probabilité\ a\ priori}{Evidence}.$$

Le terme $P(\mathcal{D}|\mathcal{H}_i)$ sert à normaliser la quantité $P(\mathcal{D}|w, \mathcal{H}_i)P(w|\mathcal{H}_i)$ et est généralement ignoré durant le processus d'apprentissage des poids du PMC. Idéalement, la solution issue de l'apprentissage devrait maximiser la probabilité à posteriori des paramètres $P(w|\mathcal{D}, \mathcal{H}_i)$ connaissant les données, et le modèle $\mathcal{H}_i$. Ce processus est désigné premier niveau d'inférence.

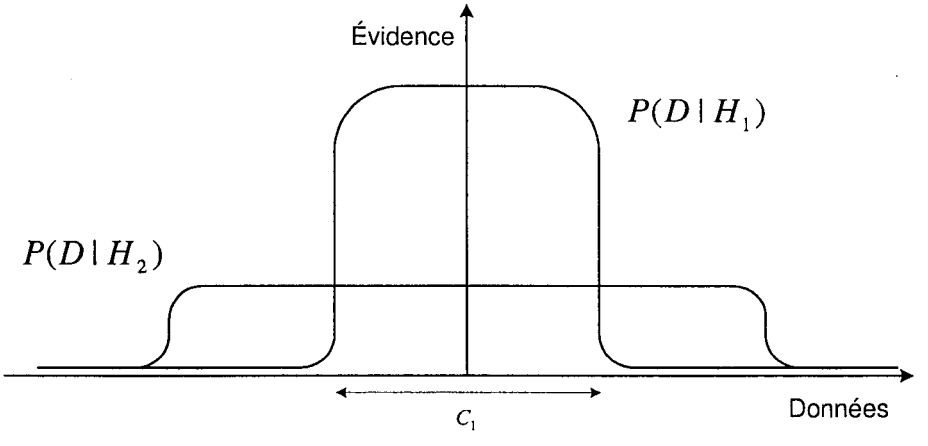


Figure 14 Sélection de modèle par l'évidence

Le second niveau d'inférence est le processus de comparaison de modèles en vue de sélectionner un classifieur optimal parmi plusieurs possibles. La meilleure approche consiste à trouver le modèle le plus plausible connaissant les données. Dans ce cas, il est important de tenir compte de l'évidence $P(\mathcal{D}|\mathcal{H}_i)$ du modèle $\mathcal{H}_i$.

Par ailleurs, la probabilité a posteriori de chaque modèle peut être approximée par :

$$P(\mathcal{H}_i|\mathcal{D}) \propto P(\mathcal{D}|\mathcal{H}_i)P(\mathcal{H}_i), \tag{4.4}$$

où $P(\mathcal{D})$ est une constante indépendante de $\mathcal{H}_i$ qui est omise. Or, étant donné que nous n'avons pas de raison de préférer un modèle $\mathcal{H}_i$ par rapport à un autre, les probabilités a priori $P(\mathcal{H}_i)$ sont choisies égales. Donc, les hypothèses $\mathcal{H}_i$ sont classées selon leurs évidences. Bien évidement, si pour une raison ou une autre, nous avons une préférence parmi les modèles disponibles, le terme $P(\mathcal{H}_i)$ de l'équation 4.4 doit être pris en compte.

Dans ce même contexte, MacKay [62] démontre que l'évidence donnée par :

$$P(\mathcal{D}|\mathcal{H}_i) = \int P(\mathcal{D}|w, \mathcal{H}_i)P(w|\mathcal{H}_i)dw \tag{4.5}$$

prend en compte le facteur d'Occam en approximant l'équation 4.5 par :

$$P(\mathcal{D}|\mathcal{H}_i) \approx P(\mathcal{D}|w_{MP}, \mathcal{H}_i)P(w_{MP}|\mathcal{H}_i)\Delta w. \tag{4.6}$$

$P(\mathcal{D}|w_{MP}, \mathcal{H}_i)$ dans l'équation 4.6 quantifie la vraisemblance des données pour les paramètres les plus probables w_{MP} du modèle (résultat de l'apprentissage), alors que $P(w_{MP}|\mathcal{H}_i)\Delta w$ représente le facteur d'Occam. Ce dernier est une quantité qui pénalise les modèles complexes au profit de modèles plus simples.

Sigurdur et al. [94], utilisèrent ce formalisme pour proposer un schéma d'optimisation de deux coefficients du «weight decay». Le premier, α^I, pénalise les poids des neurones de la couche cachée, alors que le deuxième, α^H, pénalise les poids des neurones de la couche de sortie. Sous hypothèse que la vraisemblance et la probabilité a priori des poids suivent une distribution Gaussienne, le négatif du logarithme de l'évidence est approximé par :

$$-log(\mathcal{D}|\mathcal{H}_i) \approx \beta N E_{emp}(w) + \alpha^I |w^I|^2 + \alpha^H |w^H|^2 - \frac{(N-m)\log\beta}{2} - \frac{m^I \log\alpha^I + m^H \log\alpha^H}{2} + \frac{\log|J(w)|}{2} + \frac{N\log\pi + m(\log N - \log 2)}{2} \quad . \quad (4.7)$$

m est le nombre total de paramètres du PMC. m^H et m^I représentent respectivement le nombre de poids de la couche de sortie et le nombre de poids de la couche cachée. w^H et w^I représentent respectivement le vecteur de poids de la couche de sortie et le vecteur de poids de la couche de cachée. β représente l'inverse de la variance du bruit des données et N est égal à la taille de l'ensemble de données. Le critère est minimisé moyennant une méthode de descente de gradient.

4.2.4 Approche par minimisation du risque structurel

A la base, la minimisation du risque structurel est l'algorithme d'induction utilisé pour entraîner le SVM [119]. Comme nous l'avons vu dans le chapitre précédent, cette technique sert à minimiser la capacité et l'erreur empirique du classifieur.

Vapnik quantifia la capacité des machines d'apprentissage par la dimension VC [119]. Pour un SVM, cette dernière dépend inversement de la marge qui séparent les classes. En outre, on démontre que cette quantité est proportionnelle à la distribution des données dans

l'espace augmenté F. En particulier, il est démontré que la dimension VC est proportionnelle au rapport entre le rayon R de l'hyper-sphère englobant les données et la marge de séparation des classes. Ce critère sera abordé ultérieurement dans la partie expérimentale pour des fins de comparaison avec l'erreur empirique.

4.3 Critères d'optimisation du SVM

Idéalement, nous voudrions choisir les valeurs des hyper-paramètres qui minimisent la vraie erreur du classifieur. Malheureusement, cette quantité n'est pas connue. Néanmoins, il est possible dans certains cas de l'estimer ou de trouver des approximations pessimistes à cette erreur. Dans ce qui suit, nous présentons quelques unes parmi les mesures les plus utilisées à cette fin.

4.3.1 Ensemble de validation

Comme nous l'avons vu dans la section 4.2.1, il est possible d'utiliser une partie des données pour estimer l'erreur de généralisation d'un classifieur, spécialement lorsque nous disposons d'un ensemble de données suffisamment grand. Dans ce cas ci, l'erreur sur l'ensemble de validation constitue un estimé non biaisé de l'erreur qui peut être utilisé pour optimiser le classifieur. La variance de cet estimé est d'autant plus petite que l'ensemble de validation est grand. Cette erreur s'écrit :

$$T = \frac{1}{N} \sum_{i} \Psi(-y_i f(x_i)), \tag{4.8}$$

où Ψ est la fonction échelon définie : $\Psi = 1$ si $x > 0$ et $\Psi = 0$ sinon.

Si par ailleurs, on ne dispose pas suffisamment de données pour y retrancher un ensemble de validation, il est possible de diviser l'ensemble de données en plusieurs partitions desquelles on utilise uniquement une seule partition pour l'évaluation de l'erreur et le reste

des partitions pour l'apprentissage. En répétant la procédure sur l'ensemble des partitions disponibles, la moyenne des erreurs ainsi calculées est l'erreur de validation croisée qui peut être utilisée pour évaluer la performance du classifieur. Le cas extrême où la partition de test contient une seule donnée est désignée «Leave-One-Out». Dans ce cas, l'estimation de l'erreur est la plus fidèle possible de l'erreur de généralisation.

4.3.2 Nombre de vecteurs de support

Bien que la procédure «Leave-One-Out» (LOO) permet une estimation non biaisée et précise de l'erreur, cette méthode est très coûteuse en temps de calcul. Cependant, il est plus judicieux de dériver des approximations ou des limites supérieures à cette estimée. Dans la procédure LOO, le nombre d'erreurs de classification à travers l'ensemble d'observations $\{(x_i, y_i)\}_{1 \leq i \leq l}$ est égal à :

$$\sum_{p=1}^{l} \Psi(-y_p f^p(x_p)) = \Psi(-y_p f^0(x_p) + y_p(f^0(x_p) - f^p(x_p))), \tag{4.9}$$

où f_p est la fonction de décision du SVM entraîné en excluant l'exemple x_p de l'ensemble d'apprentissage et f_0 est la fonction de décision du SVM entraîné sur la totalité des données d'apprentissage. Il est démontré que dans le cas séparable, le terme de l'équation 4.9 est bornée par le nombre de vecteurs de support $\sharp SV$ [115]. D'où la limite supérieure :

$$T = \frac{\sharp SV}{l}, \tag{4.10}$$

sur l'erreur de l'estimateur LOO. Celle-ci a été proposée par Vapnik puis utilisée par Scholkopf pour caractériser la complexité du SVM [87]. Ce critère reste toutefois une borne très pessimiste de l'erreur de généralisation et ne constitue donc pas une mesure fiable de la performance du classifieur.

4.3.3 Borne de Jaakkola-Haussler

Jaakola et al. [45] étaient les premiers à constater que lors d'une procédure «Leave-One-Out» et pour un SVM sans biais, l'inégalité suivante est toujours vérifiée :

$$y_p(f^0(x_p) - f^p(x_p)) \leq \alpha_0^p K(x_p, x_p).$$

D'où la borne de Jaakola [45] s'écrivant :

$$T = \frac{1}{l}\sum_{p=1}^{l} \Psi(\alpha_0^p K(x_p, x_p) - 1).$$

4.3.4 Borne de Opper-Winther

Opper et al. [72] se sont inspirés de la théorie de la réponse linéaire pour prouver que dans le cas du SVM sans biais en présence de données non séparables, nous avons :

$$y_p(f^0(x_p) - f^p(x_p)) = \frac{\alpha_0}{(K_{SV}^{-1})_{pp}},$$

où K_{SV} est la matrice de produit scalaire des vecteurs de support. Ce qui donne la borne :

$$T = \frac{1}{l}\sum_{p=1}^{l} \Psi(\frac{\alpha_0}{(K_{SV}^{-1})_{pp}} - 1).$$

4.3.5 Dimension VC

Les travaux de Vapnik sur le risque structurel ont permis de dériver une borne pessimiste de l'erreur de généralisation pour le cas du SVM sans biais (dont l'hyper-plan passe par l'origine) et en présence de données séparables [120]. Cette borne est la dimension de Vapnik-Chernovenkis déjà abordée dans le paragraphe 4.2.4. Cette dernière est donnée

par :

$$T = E\{\frac{1}{l}\min(\frac{R^2}{\gamma^2}, l)\},$$

où R représente le rayon de la plus petite hyper-sphère englobant les données dans l'espace augmenté F du noyau, γ est la marge de séparation des classes, et l est la taille de l'ensemble d'apprentissage. E représente l'espérance mathématique à travers les données. En pratique, la communauté de chercheurs considèrent plutôt l'approximation :

$$\frac{R^2}{\gamma^2},$$

appelé critère *«rayon-marge»* [23]. Nous reviendrons sur ce critère avec plus de détails dans la partie méthodologique.

4.3.6 Borne de Joachim

Joachims [47] proposa un estimé analytique pessimiste de l'erreur de validation croisée LOO. Cette approximation est fonction des multiplicateurs α_i des vecteurs de support et des variables ressort ξ_i correspondant aux données se trouvant à l'intérieur ou au delà de la marge.

Elle s'écrit :

$$T = \frac{1}{l} card\{i : 2\alpha_i R_\delta^2 + \xi_i \geq 1\}.$$

R_δ^2 est un réel satisfaisant $c \leq k(x_i, x_j) \leq c + R_\delta^2$ pour une constante c et $\forall i, j$.

4.3.7 Portée des vecteurs de support («Span bound»)

Chapelle et Vapnik [23, 118] proposent une borne sur l'erreur LOO dérivée du concept de portée des vecteurs de support. Cette quantité est établie sous l'hypothèse que les vecteurs

de support demeurent inchangés pendant la validation croisée LOO. D'où l'égalité :

$$y_p(f^0(x_p) - f^p(x_p)) = \alpha_p^0 S_p^2,$$

où S_p représente la distance entre l'image $\Phi(x_p)$ de x_p et le point Λ_p défini par :

$$\Lambda_p = \{ \sum_{i \neq p, \alpha_i^0} \lambda_i \Phi(x_i), \sum_{i \neq p} \lambda_i = 1 \}. \tag{4.11}$$

Ceci permet d'exprimer le rapport du nombre d'erreurs de classification à la taille des données par :

$$T = \frac{1}{l} \sum_{p=1}^{l} \Psi(\alpha_p^0 S_p^2 - 1). \tag{4.12}$$

4.3.8 Erreur de Validation Croisée Généralisée- GACV

Quelques travaux démontrent le lien existant entre la formulation du SVM et la régularisation des machines d'apprentissage [122]. En particulier, Wahba et al. [123] montrèrent la similitude entre la maximisation de la marge du SVM et la régularisation d'une fonction objective via la pénalisation du principe de maximum de vraisemblance. L'auteur aborde aussi la problématique du choix des hyper-paramètres du SVM dans le contexte des RKHS (de l'anglais «Reproducing Kernel Hilbert Space») en présentant un nouvel estimé de l'erreur de généralisation appelée GACV (de l'anglais «Generalized Approximate Cross-Validation»). A l'origine, ce travail a été entrepris par l'auteur afin d'optimiser le compromis «biais-variance» du SVM en présence de données non séparables [125]. La méthode est basée sur un estimé pessimiste de l'erreur de généralisation dérivé du GCKL (de l'anglais «Generalized Comparative Kullback Leibler Distance»).

L'expression du GACV s'écrit :

$$GACV(\underline{\alpha}, \underline{\theta}) = \frac{1}{l}\left(\sum_{i=1}^{l}\xi_i + 2\sum_{y_i f_i < -1}\alpha_i K_{ii} + \sum_{y_i f_i \in [-1,1]}\alpha_i K_{ii}\right); \tag{4.13}$$

où ξ_i représente la variable ressort associée à l'observation x_i définie :

$$\xi_i = 0 \quad si\ y_i f_i \geq 1$$
$$\xi_i = 1 - y_i f_i \quad si\ y_i f_i < 1,$$

et où K_{ii} représente la valeur du noyau $K(x_i, x_i)$. Ce critère sera considéré plus loin dans le chapitre 5.

4.4 Conclusion

Dans ce chapitre, nous discutons de la problématique de sélection de modèle en général. La sélection des hyper-paramètres des classifeurs y est présentée comme un problème d'optimisation qui doit satisfaire le compromis «biais-variance». De par les travaux considérés, nous mettons en relief le lien entre les différentes approches de régularisation et la sélection de modèle.

En particulier, nous nous inspirons des travaux sur les réseaux de neurones pour montrer la complexité et la subtilité de la théorie. Enfin, dans la dernière partie, nous donnons quelques uns des critères de sélection de modèle pour le SVM. Plus particulièrement, la dimension VC et GACV sont deux critères algébriques populaires que nous considérerons dans les parties méthodologique et expérimentale.

CHAPITRE 5

OPTIMISATION DES HYPER-PARAMÈTRES DU SVM : CAS BICLASSE

Cette partie est consacrée à la problématique de sélection de modèle pour le SVM en classification. Dans ce qui suit, nous justifions l'importance du choix des hyper-paramètres sur la performance du classifieur. Aussi, nous situons la problématique dans le contexte spécifique du SVM pour un problème biclasse. Plus loin, nous présentons trois schémas d'optimisation automatique des hyper-paramètres. Le premier est inspiré des travaux de Vapnik sur le risque structurel. Cette méthode a été formalisée par Chapelle et al. [24].

En deuxième lieu, nous présentons une nouvelle méthodologie basée sur la minimisation automatique d'une erreur empirique estimée sur un ensemble de validation [7, 9]. Ce critère permet une estimation plus fiable et moins pessimiste de l'erreur.

En dernier lieu, nous présentons une méthode basée sur la minimisation d'un critère analytique approximant l'erreur de validation croisée. Le GACV (de l'anglais «Generalized Approximate Cross Validation») est une estimation du premier ordre de l'erreur LOO («Leave-One-Out») [124] pour laquelle nous proposons un nouvel algorithme de minimisation.

Pour chacune des méthodes, nous expliquons le critère et décrivons son algorithme de minimisation. Pour l'erreur empirique, nous présentons une version rapide de l'algorithme de minimisation pour accélérer la convergence sur de gros ensembles de données.

5.1 Introduction

Soit un noyau K fonction d'un ou de plusieurs paramètres encodés dans un même vecteur $\underline{\theta} = (\theta_1, ..., \theta n)$. Le SVM modélise la classes de fonctions conditionnées par $\underline{\alpha}$, b et $\underline{\theta}$ selon

l'équation :

$$f_{\underline{\alpha},b,\underline{\theta}} = \sum_i \alpha_i y_i K_\theta(x, x_i) + b. \tag{5.1}$$

Dans ce qui suit, $\underline{\alpha}$ représente le vecteur des multiplicateurs α_i associés aux couplets de données (x_i, y_i), où x_i et y_i représentent respectivement le vecteur d'entrée et son étiquette. Sa taille est égale au nombre de vecteurs de support.

Dans l'état de l'art, nous avons discuté longuement des approches existantes d'optimisation d'hyper-paramètres. Celles-ci dépendent à la fois du critère considéré et du classifieur utilisé. Aussi, comme nous l'avons déjà évoqué, parmi les paramètres de noyaux influençant notablement la performance du SVM, on citera le paramètre σ du RBF, le triplet (a, b, d) du noyau polynômial, le couplet (a, b) de la sigmoïde et le couplet (γ, σ) de KMOD (Tableau VII).

Tableau VII

Noyaux de Mercer

noyaux	**expression**
linéaire	$K(x,y) = x.y$
sigmoïde	$K(x,y) = \tanh(ax.y + b)$
polynômial	$K(x,y) = (b + ax.y)^d$
gaussien	$K(x,y) = \exp(-\frac{\|x-y\|^2}{\sigma^2})$
laplacien	$K(x,y) = \exp(-\|\gamma\|\|x-y\|)$
KMOD	$K(x,y) = a[\exp(\frac{\gamma^2}{\|x-y\|^2+\sigma^2}) - 1]$

La variable C du SVM, est un hyper-paramètre additionnel qui influence la généralisation du classifieur lorsque les classes sont fortement enchevêtrées où un bruit considérable subsiste.

5.2 Minimisation de la dimension VC

La dimension VC d'un classifieur (dimension de Vapnik-Chernovenkis) déjà abordée dans la section 4.2.4, représente le nombre maximal d'exemples séparables selon toutes les combinaisons possibles des étiquettes (1 ou -1). Cette quantité est aussi connue sous le vocable «fat shattering». Il a été démontré, par exemple, que la dimension VC d'un classifieur linéaire est au moins égale à la dimension $n + 1$ des données [21].

Pour le SVM non linéaire, les travaux de Vapnik sur le risque structurel ont permis de dériver une borne supérieure de la dimension VC [117]. Dans le cas de données séparables, cette quantité s'écrit :

$$h \leq R^2 \|w\|^2 + 1, \tag{5.2}$$

où R représente le rayon de la plus petite sphère englobant les données d'apprentissage dans l'espace augmenté F, et $\|w\|$ représente la norme du vecteur normal à l'hyper-surface de séparation du SVM défini dans le chapitre 3.

L'expression du risque structurel dans les équations 3.5 et 3.6, suggère que la minimisation de h connaissant l réduit la complexité du classifieur. Ceci est rendu possible en réduisant à la fois $\|w\|$ et R, ou en minimisant le rapport du rayon R à la marge. Dès lors, la marge seulement ne peut pas être un critère suffisant pour l'optimisation.

Quelques travaux ont utilisé ce critère sur certaines données. Ainsi, Scholkopf [87] analyse la dimension VC afin de choisir le degré du noyau polynômial. Cependant, si le noyau contient plus d'un paramètre, la procédure nécessite le calcul du critère pour plusieurs plages de valeurs. Ceci est fastidieux et ne garantit pas d'optimum global. Chapelle et al. [24] proposent un formalisme d'optimisation automatique des paramètres du noyau et du paramètre C (pour le SVM L2) basé sur la minimisation du rapport entre le rayon R et la marge. L'auteur analyse les résultats de la méthode sur certains ensembles de données de UCI [16]. Nous présentons dans ce qui suit l'algorithme de minimisation du critère.

5.2.1 Algorithme

Le rayon de la plus petite hyper-sphère englobant les données d'apprentissage est trouvé en considérant la fonction objective :

$$R^2(\beta) = \max_\beta \sum_{i=1}^{l} \beta_i K(x_i, x_i) - \sum_{i=1}^{l} \sum_{j=1}^{l} \beta_i \beta_j K(x_i, x_j) \quad (5.3)$$

$$\texttt{sujet à :} \quad \sum_{i=1}^{l} \beta_i = 1,$$

$$\forall i \; \beta_i \geq 0,$$

où K représente le noyau courant du SVM. Si aucun noyau n'est utilisé, l'hyper-sphère est alors calculée dans l'espace d'entrée. La figure 15 représente l'hyper-sphère englobant des données à deux dimensions avec un noyau linéaire.

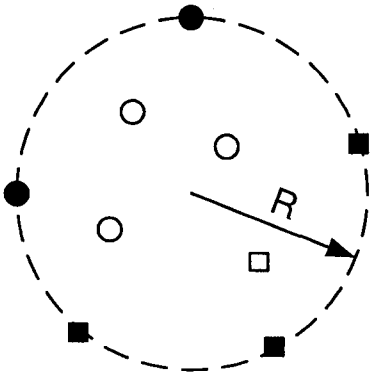


Figure 15 Données 2D et cercle englobant

La solution de la fonction objective de ci-dessus est l'ensemble des variables β_i associées aux vecteurs d'entrée x_i. L'hyper-sphère est alors caractérisée par le sous-ensemble des données dont les β_i correspondants sont différents de zéro. Tous les points situés à l'intérieur de l'hyper-sphère ont un β_i égal à zéro.

Dans le but de minimiser le critère, il est plus simple d'utiliser une descente de gradient en estimant les gradients de R^2 et de $\|w\|^2$ par rapport aux hyper-paramètres. Ainsi, pour

minimiser la dimension VC, il suffit de considérer le critère «rayon-marge» [1] donné par $T = R^2\|w\|^2$. La figure 16 montre la variation de la dimension VC en fonction des paramètres γ et σ de KMOD.

Soit θ un hyper-paramètre à optimiser. Le gradient de T par rapport à θ s'écrit :

$$\frac{\partial T}{\partial \theta} = R^2 \frac{\partial \|w\|^2}{\partial \theta} + \|w\|^2 \frac{\partial R^2}{\partial \theta}$$

D'après l'équation 5.3, le gradient du carré du rayon peut s'écrire :

$$\frac{\partial R^2}{\partial \theta} = \sum_{i=1}^{l} \beta_i \frac{\partial K(x_i, x_i)}{\partial \theta} - \sum_{i=1}^{l} \sum_{j=1}^{l} \beta_i \beta_j \frac{\partial K(x_i, x_j)}{\partial \theta} \tag{5.4}$$

[1] Différenciant ainsi la dimension VC de sa borne supérieure

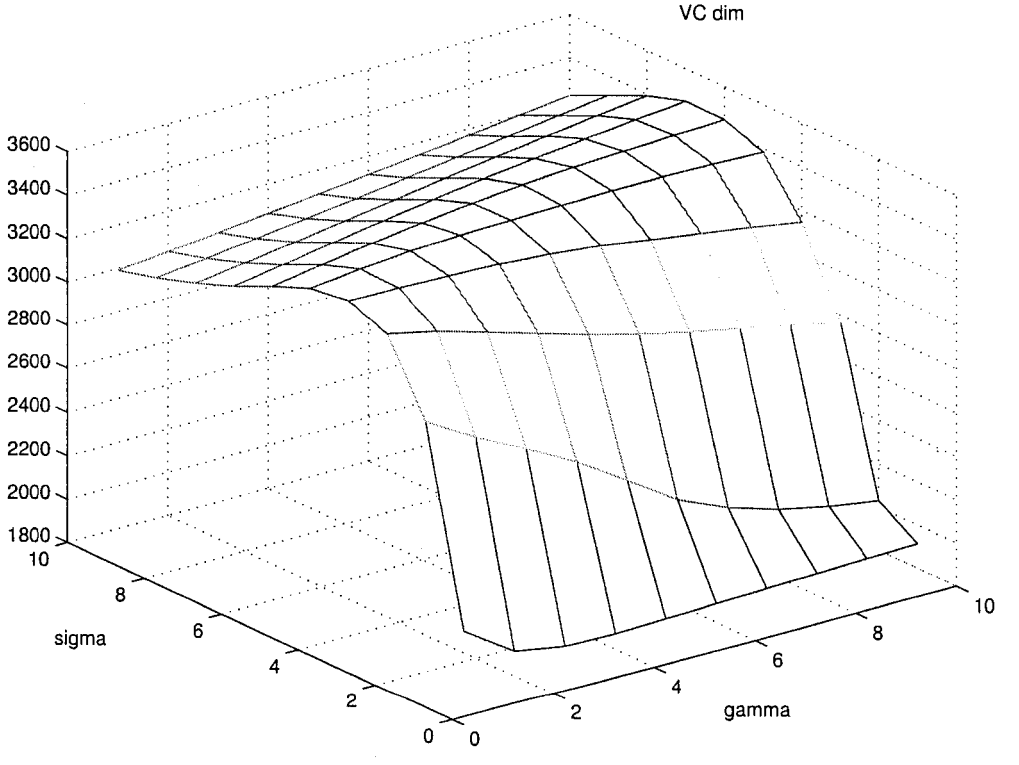


Figure 16 Variation de la dimension VC en fonction des paramètres de KMOD

Par ailleurs, il est démontré que l'expression de $\|w\|^2$ est égale à $2W(\alpha^0)$, où $W(\alpha^0)$ est égal à la valeur de l'objective duale décrite dans l'équation 3.27. L'expression du gradient de $\|w\|^2$ s'écrit alors :

$$\frac{\partial \|w\|^2}{\partial \theta} = -\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \frac{\partial K(x_i, x_j)}{\partial \theta}.$$

Les variables y_i et α_i représentent respectivement l'étiquette et le multiplicateur de Lagrange associés à l'exemple x_i. Une description de l'algorithme d'optimisation par minimisation de la dimension VC est comme suit :

Algorithme

1) Initialiser $\underline{\theta}$ à $\underline{\theta}_0$,

2) tant que pas convergence :

   a) apprentissage du SVM pour $\underline{\theta}$,

   b) calculer $\frac{\partial T}{\partial \underline{\theta}}$,

   c) corriger $\underline{\theta}$ selon : $\Delta\underline{\theta} = -\eta \frac{\partial T(\underline{\alpha}, \underline{\theta})}{\partial \underline{\theta}}$,

3) apprentissage du SVM avec $\underline{\theta}$ final,

La variable η représente le pas d'apprentissage dont la valeur sert à régler l'amplitude $\Delta\underline{\theta}$.

5.3 Minimisation de l'Erreur Empirique

L'utilisation de la dimension VC comme critère de sélection de modèle telle que présenté précédemment impose quelques remarques fort importantes avant d'envisager son implantation dans un système effectif de reconnaissance de formes.

- Le calcul de $\|w\|^2$ est de complexité $O(V^2)$; V étant le nombre de vecteurs de support,
- le calcul de R^2 est un problème d'optimisation quadratique dont la résolution est aussi coûteuse que l'apprentissage du SVM. Son calcul est de complexité au moins

égale à $O(N^2)$. Pour un ensemble de quelques milliers d'exemples, sa complexité est de l'ordre de $O(N^3)$; N étant la taille de l'ensemble d'apprentissage.

- le critère peut nécessiter une normalisation des données dans l'espace augmenté F [24],
- la dimension VC est une borne supérieure qui peut s'avérer être un estimé très biaisé de l'erreur de généralisation.

Il est préférable de disposer d'un critère fournissant une approximation assez fiable de l'erreur de généralisation et moins complexe à calculer. Ceci ne peut être garanti qu'en considérant des données du même problème.

L'utilisation de l'ensemble des données disponibles en validation croisée est une méthode bien connue. L'inconvénient reste toutefois sa complexité de calcul. Comme décrit dans la section 4.2.1, la méthode nécessite plusieurs passes d'apprentissage et de test. La cas extrême est lorsqu'on considère autant de passes que de données disponibles. L'estimé de l'erreur de généralisation est alors quasiment non biaisée. Néanmoins, si l'on dispose d'un grand nombre d'exemples (quelques milliers d'exemples), il est possible d'en retrancher une partition de validation qui servira à calibrer le système. Les données restantes sont à départager entre les ensembles d'apprentissage et de test. Il est démontré que la précision de l'estimé de l'erreur sur l'ensemble de validation est d'autant plus grande que la taille des données est importante. On démontre que la variance de l'estimé de l'erreur s'annule si la taille des données de validation est infinie.

Des méthodes d'optimisation automatique d'hyper-paramètres sur la base d'un ensemble de validation ont déjà été étudiées dans le contexte d'autres classifieurs. Ainsi, Larsen et al. dans [53, 94] présentent une méthode d'optimisation du paramètre de régularisation du «weight decay» en utilisant le critère «OBD-Optimal Brain Damage» estimé sur des données de validation. Aussi, Bengio dans [13], présente un formalisme utilisant l'estimé

de l'erreur quadratique sur l'ensemble de validation afin optimiser les hyper-paramètres d'un perceptron multicouche en régression.

Nous présentons dans ce qui suit, une nouvelle méthodologie d'optimisation des hyper-paramètres du SVM basée sur l'estimation de l'erreur empirique. Cette dernière est une approximation de l'erreur de généralisation calculée sur la base d'un ensemble de validation de taille finie et supposé de même distribution que l'ensemble des données d'apprentissage et de test. La figure 17 donne un aperçu du schéma de l'optimisation proposée.

5.3.1 Critère d'optimisation

Soit $f(x)$ la fonction de décision du SVM. Le nombre d'erreurs de classification sur l'ensemble de validation s'écrit :

$$T = \frac{1}{N} \sum_{i} \Psi(-y_i f(x_i)) \tag{5.5}$$

(où Ψ est une fonction de Heaviside et N est la taille de l'ensemble de validation). Cette quantité reflète le pouvoir de généralisation du classifieur. De fait, elle constitue un critère d'optimisation dont la minimisation optimise la généralisation du classifieur. Or, la fonction échelon Ψ n'est pas dérivable. Ce critère ne peut donc pas être utilisé conjointement avec une méthode de descente de gradient pour une optimisation automatique de paramètres.

Soit maintenant une observation x_i de l'ensemble de validation. Et soient t_i et y_i respectivement l'étiquette binaire (0 ou 1) et bipolaire (+1 ou -1) de la donnée telle que $t_i = 0$ corresponde à $y_i = -1$ et $t_i = 1$ corresponde à $y_i = 1$. On a donc :

$$t_i = \frac{y_i + 1}{2}.$$

Soit maintenant :

$$z_i = \frac{[f_i] + 1}{2}$$

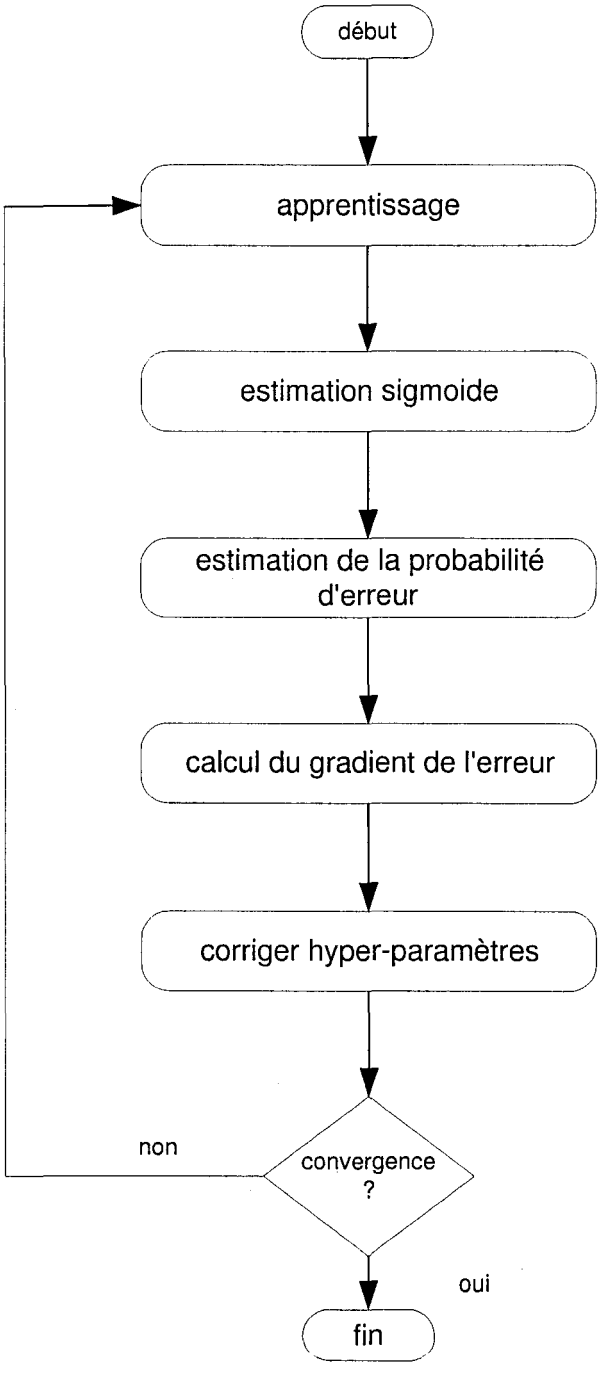


Figure 17 Schéma représentant les étapes de l'optimisation avec réduction de l'erreur empirique

l'étiquette binaire inférée par le classifieur, tel que $f_i = f(x_i)$ est la sortie du SVM pour l'observation x_i. L'opérateur [ ] représente la fonction signe définie :

$$[x] = +1 \; si \; x > 0$$
$$[x] = -1 \; si \; x \leq 0. \tag{5.6}$$

Supposons maintenant que p_i est la probabilité à posteriori $P(y = +1|x_i)$ que l'observation x_i soit de la classe positive. Idéalement, on peut espérer que les données de validation appartenant à la classe positive aient une probabilité à posteriori $p_i = 1$, et que les données de validation de classe négative ait une probabilité à posteriori $p_i = 0$.

Il est donc possible de caractériser la probabilité d'erreur E_i d'un exemple x_i de la classe positive par la quantité $1 - p_i$, et celle d'un exemple x_i de la classe négative par p_i. L'erreur résiduelle E_i peut encore être formulée par l'équation :

$$E_i = P(y_i \neq z_i) = p_i^{1-t_i}(1 - p_i)^{t_i}, \tag{5.7}$$

où t_i représente l'étiquette binaire de l'observation x_i. Pour un ensemble de validation de taille N, la moyenne des erreurs peut s'écrire :

$$E = \frac{1}{N}\sum_{i=1}^{N} E_i = \frac{1}{N}\sum_{i=1}^{N} p_i^{1-t_i}(1 - p_i)^{t_i}. \tag{5.8}$$

Dans l'hypothèse où p_i représente la vraie probabilité à posteriori associée à l'observation x_i, et que la taille de l'ensemble de validation est infinie ($N = \infty$), la valeur de E donnée ci-haut tend vers l'erreur de classification naturelle.

Étant donnée l'estimation $\hat{p}_i$ de p_i, il est possible de considérer l'estimé de l'erreur :

$$E = \frac{1}{N}\sum_{i=1}^{N} E_i = \frac{1}{N}\sum_{i=1}^{N} \hat{p}_i^{1-t_i}(1-\hat{p}_i)^{t_i} \tag{5.9}$$

comme étant une approximation raisonnable de l'erreur de généralisation actuelle du classifieur. En outre, en vue d'optimiser le SVM, on pourrait tenter de minimiser cette erreur.

Par ailleurs, E_i peut s'écrire $|t_i - \hat{p}_i|$ qui est similaire au degré près à l'expression de l'erreur quadratique utilisée dans l'apprentissage du PMC et d'un certain nombre de classifieurs. Aussi, cette erreur est un cas particulier de l'erreur de Minkowski-R définie :

$$|t_i - \hat{p}_i|^R,$$

où R est un entier supérieur ou égal à 1. Pour $R = 1$, l'erreur est de norme L1, et on démontre qu'elle équivaut au négatif du logarithme de la vraisemblance des données étant donné un bruit Laplacien sur les étiquettes de la forme :

$$p(\epsilon) = a\exp(-\beta|\epsilon|),$$

a, β représentent respectivement une constante de normalisation, et l'inverse de la variance de la distribution de l'erreur. Hanson et al. mentionnent que l'erreur L1 a l'avantage de pénaliser les observations marginales présentant des erreurs relativement grandes contrairement à l'erreur L2 (erreur quadratique) qui magnifie l'effet de ces observations [39]. Par ailleurs, on démontre que la minimisation de l'erreur L1 (R=1) estime la valeur médiane conditionnelle des données contrairement à l'erreur L2 (R=2) qui estime la valeur moyenne conditionnelle des données [39, 15].

Afin de pouvoir calculer l'estimé de la probabilité de l'erreur, nous décrivons la procédure d'estimation de la probabilité à posteriori utilisée dans l'équation 5.9 donnée plus haut.

5.3.2 Estimation de la probabilité à posteriori

L'estimation de la probabilité à partir du score brut fourni par le SVM peut s'avérer une tâche complexe à réaliser.

Sollich et al. proposa une méthode basée sur la théorie Bayesienne afin de calculer la probabilité à posteriori et de sélectionner les hyper-paramètres [62, 100]. Son formalisme interprète le SVM comme une maximisation de la probabilité à posteriori du modèle sachant les données sous hypothèse que la probabilité à priori des hyper-paramètres du modèle respecte une distribution gaussienne.

Par ailleurs, Wahba utilisa une fonction logistique de la forme :

$$P(y = +1|x) = \frac{1}{1 + \exp(-f(x))},$$

où $f(x)$ est la sortie brute du SVM et y représente l'étiquette de l'observation x. Cette fonction est connue sous le nom de fonction sigmoïde, et est utilisée comme fonction logistique dans le perceptron multicouche [124]. Dans le même ordre d'idées, Platt propose dans [76] une fonction logistique à deux paramètres de la forme :

$$P(y = 1|f) = \frac{1}{1 + \exp(Af + B)}. \tag{5.10}$$

Ce modèle est facile à calculer et requiert une procédure d'optimisation non-linéaire du couple de paramètres (A, B). La constante B dans l'équation 5.10 permet un biais pour le seuil de Bayes optimal de telle manière qu'une probabilité de sortie de 0.5 puisse correspondre à une valeur de sortie non nulle du SVM.

Le choix des valeurs de A et B est fait en minimisant l'entropie croisée des données qui s'écrit :

$$entropie = -\sum_i t_i \log(p_i) + (1 - t_i) \log(1 - p_i); \tag{5.11}$$

où la somme est réalisée sur l'ensemble des observations de validation. Dans l'équation 5.11,

$$\hat{p}_i = \frac{1}{1 + \exp(Af_i + B)}$$

représente la probabilité à posteriori estimée.

Notons que la fonction décrite dans l'équation 5.10 permet d'estimer la probabilité de l'erreur sur l'ensemble de validation qui, par ailleurs, est dérivable sous contrainte que la sortie brute du SVM soit dérivable aussi par rapport au paramètre à optimiser. Nous nous affranchissons ainsi de l'erreur de classification donnée dans l'équation 5.5.

Afin d'identifier les paramètres A et B de la sigmoïde, nous utiliserons une variante de l'algorithme de Newton pour la minimisation de l'entropie croisée donnée précédemment [76, 58].

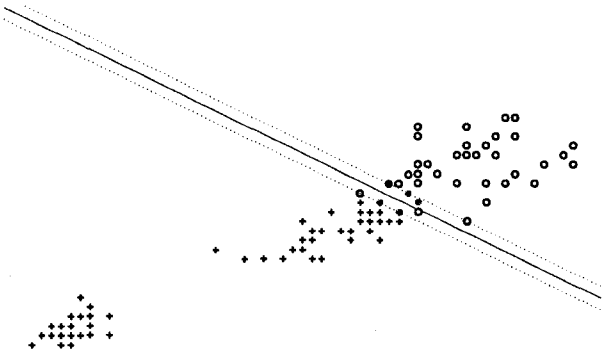
La figure 18 montre respectivement la frontière de séparation et la fonction logistique inférée pour un SVM linéaire sur les données iris3v12 [16]. Ci-dessous, nous intégrons la procédure dans le processus d'optimisation comme décrit dans ce qui suit.

5.3.3 Algorithme

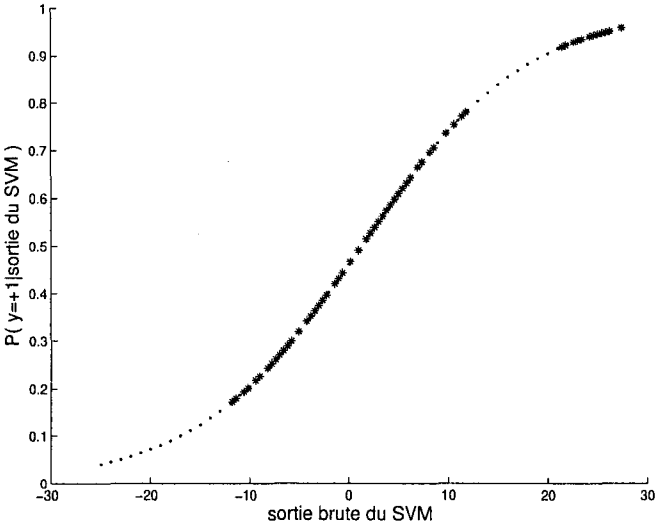
La fonction objective que nous proposons sert à optimiser les hyper-paramètres du classifieur. Dans le but de minimiser la probabilité de l'erreur nous utilisons un algorithme de descente de gradient. Si la fonction objective est convexe [2], la descente de gradient est garantie de converger vers un minimum. En négligeant l'information de second ordre, et en considérant le proche voisinage du minimum, nous pouvons approximer l'expression du gradient par rapport aux hyper-paramètres $\underline{\theta}$ comme suit :

$$\nabla E(\underline{\theta}) = \frac{\partial E}{\partial \underline{\theta}} = \frac{\partial E}{\partial \underline{\alpha}} \frac{\partial \underline{\alpha}}{\partial \underline{\theta}} \tag{5.12}$$

[2] Sous-entend qu'elle soit monotone



(a) Frontière de décision du SVM avec un noyau linéaire



(b) Estimation de la probabilité à posteriori

Figure 18 Sigmoïde inférée sur les données Iris (classe 3 contre 1 et 2)

où $\underline{\alpha} = (\alpha_1, ..., \alpha_k)$ représente le vecteur des multiplicateurs α_i, k étant égal au nombre de vecteurs de support.

Le gradient de l'erreur par rapport aux hyper-paramètres est un vecteur de dérivées partielles de taille n égale au nombre d'hyper-paramètres du classifieur, et s'écrit :

$$\nabla E(\underline{\theta}) = \frac{\partial E}{\partial \underline{\theta}} = \left(\frac{\partial}{\partial \theta_1} E(\underline{\alpha}, \underline{\theta}), \ldots, \frac{\partial}{\partial \theta_n} E(\underline{\alpha}, \underline{\theta}) \right) \tag{5.13}$$

Le gradient de l'erreur par rapport aux multiplicateurs est un vecteur de dérivées partielles de taille k égale au nombre de vecteurs de support, et s'écrit :

$$\nabla E(\underline{\alpha}) = \frac{\partial E}{\partial \underline{\alpha}} = \left(\frac{\partial}{\partial \alpha_1} E(\underline{\alpha}, \underline{\theta}), \ldots, \frac{\partial}{\partial \alpha_k} E(\underline{\alpha}, \underline{\theta}) \right) \tag{5.14}$$

Le gradient du vecteur des multiplicateurs $\underline{\alpha}$ par rapport aux hyper-paramètres est une matrice de dérivées partielles de taille $k \times n$, et s'écrit :

$$\nabla \underline{\alpha}(\underline{\theta}) = \frac{\partial \underline{\alpha}}{\partial \underline{\theta}} = \left(\frac{\partial \underline{\alpha}}{\partial \theta_1}, \ldots, \frac{\partial \underline{\alpha}}{\partial \theta_n} \right) = \begin{pmatrix} \frac{\partial \alpha_1}{\partial \theta_1} & \cdots & \frac{\partial \alpha_1}{\partial \theta_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \alpha_k}{\partial \theta_1} & \cdots & \frac{\partial \alpha_k}{\partial \theta_n} \end{pmatrix}. \tag{5.15}$$

Considérant maintenant les dérivées partielles $\frac{\partial}{\partial \alpha_j} E(\underline{\alpha}, \underline{\theta})$. Il est possible de décomposer le gradient de l'erreur empirique sur l'ensemble des erreurs résiduelles E_i comme suit :

$$\frac{\partial}{\partial \alpha_j} E(\underline{\alpha}, \underline{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial}{\partial \hat{p}_i} E_i(\underline{\alpha}, \underline{\theta}) \frac{\partial \hat{p}_i}{\partial f_i} \frac{\partial f_i}{\partial \alpha_j}, \tag{5.16}$$

où

$$\frac{\partial E_i}{\partial \hat{p}_i} = -\hat{p}_i^{1-t_i} t_i (1 - \hat{p}_i)^{t_i - 1} + (1 - t_i)(1 - \hat{p}_i)^{t_i} \hat{p}_i^{-t_i} = \begin{cases} +1 \; if \; t_i = 0 \\ -1 \; if \; t_i = 1 \end{cases} \tag{5.17}$$

Le gradient de la probabilité à posteriori $\hat{p}_i$ par rapport à la sortie brute f_i du SVM s'écrit :

$$\frac{\partial \hat{p}_i}{\partial f_i} = -A \hat{p}_i^2 \exp(A f_i + B) = -A \hat{p}_i (1 - \hat{p}_i). \tag{5.18}$$

Tandis que le gradient de f_i par rapport à α_j s'écrit :

$$\frac{\partial f_i}{\partial \alpha_j} = y_j K_{\underline{\theta}}(x_j, x_i), \tag{5.19}$$

où $y_j = \pm 1$ représente l'étiquette de l'observation x_j.

Une fois la dérivée de l'erreur par rapport au multiplicateurs $\underline{\alpha}$ calculée, il reste à estimer la matrice $\nabla \underline{\alpha}(\underline{\theta})$. Notons qu'il est possible d'inclure la variable du biais b dans le vecteur

$\underline{\alpha}$ tel que :

$$\underline{\alpha} = (\alpha_1, ..., \alpha_k, b).$$

En posant

$$\underline{H} = \begin{pmatrix} \underline{K}^Y & \underline{Y} \\ \underline{Y}^T & 0 \end{pmatrix},$$

tel que $K_{ij}^Y = y_i y_j K(x_i, x_j)$ et $\underline{Y}$ est le vecteur des étiquettes des vecteurs de support ($\underline{Y}^T$ étant son vecteur transposé). On démontre l'égalité suivante :

$$\nabla \underline{\alpha}(\underline{\theta}) = \frac{\partial \underline{\alpha}}{\partial \underline{\theta}} = -\underline{H}^{-1} \frac{\partial \underline{H}}{\partial \underline{\theta}} \underline{\alpha}^T, \tag{5.20}$$

dans le cas de données séparables [24].

La matrice $\underline{H}$ est une matrice de taille $(k+1) \times (k+1)$, où k est le nombre de vecteurs de support. $\underline{K}^Y$ représente la dérivée seconde de la fonction objective du SVM par rapport aux variables α_i des vecteurs de support. Nous appellerons cette matrice de Grahm-Schmidt modifiée en référence à la matrice de noyau $\underline{K}^Y$.

Ci-dessous est donné l'algorithme de minimisation de l'erreur empirique pour un problème de classification de données binaires.

Algorithme

---

1) Initialiser $\underline{\theta}$ à $\underline{\theta}_0$,

2) tant que pas convergence :

   a) apprentissage du SVM,

   b) estimer les paramètres A et B de la fonction sigmoïde,

   c) estimer la probabilité d'erreur de classification,

   d) calculer le gradient de l'erreur $\frac{\partial}{\partial \underline{\theta}} E(\underline{\alpha}, \underline{\theta})$,

   e) modifier $\underline{\theta}$ selon : $\Delta \underline{\theta} = -\eta \frac{\partial}{\partial \underline{\theta}} E(\underline{\alpha}, \underline{\theta})$,

3) apprentissage du SVM avec $\underline{\theta}$ final.

Notons que η représente le facteur d'amplitude à apporter au gradient calculé. Cette constante est équivalente au facteur d'apprentissage du PMC. Sa valeur contrôle la vitesse de convergence de l'algorithme.

5.3.4 Accélération par la méthode de Quasi-Newton

L'apprentissage du SVM est un problème d'optimisation quadratique sous contraintes. Il n'est donc pas possible d'optimiser les hyper-paramètres simultanément avec les paramètres du modèle. Néanmoins, quelques travaux récents ont effleuré ce problème avec d'autres modèles de classification. Ainsi, Vincent et al. proposèrent une version modifiée du perceptron dite «kenel-perceptron» intégrant un noyau non linéaire. Les paramètres et les hyper-paramètres de ce modèle sont estimés simultanément en minimisant l'erreur quadratique sur l'ensemble d'apprentissage [121]. Suykens et al. dans [107], utilisent une approche Bayesienne pour sélectionner les valeurs des hyper-paramètres du «LS SVM». Ce dernier est une version modifiée du SVM dont l'apprentissage minimise une erreur quadratique. Ceci contourne l'optimisation quadratique adoptée dans le SVM classique.

Notre système étant basé sur la version classique du SVM [3], il convient de découpler l'apprentissage du classifieur de l'optimisation des hyper-paramètres tel que présenté précédemment. Une limitation toutefois de la procédure est la nécessité d'entraîner le classifieur à chaque mise à jour des hyper-paramètres. En outre, la convergence de la procédure peut prendre un grand nombre d'itérations. Ceci constitue un inconvénient surtout lorsqu'on résout un grand ensemble de données. Une alternative à ceci serait d'utiliser l'information de second ordre pour accélérer la convergence de la procédure vers un minimum en beaucoup moins d'itérations. La méthode de Newton par exemple, utilise la dérivée seconde de la fonction objective pour trouver la direction de descente vers le minimum. Toutefois ceci présente quelques inconvénients :

[3] Ceci n'est pas tout a fait un inconvénient car l'on est sûr que le SVM classique minimise le risque structurel. Ceci n'est pas le cas du «LS SVM» et n'est pas vrai pour le «kernel-Perceptron»

1. l'estimation du Hessien est coûteuse en temps de calcul,
2. les dérivées secondes recherchées peuvent ne pas exister ou ne pas être dérivables,
3. le Hessien peut ne pas être défini positif.

Une alternative naturelle à cette méthode est l'algorithme du Quasi-Newton. Ce dernier, malgré son efficacité, n'a trouvé application que sur des problèmes de quelques paramètres. La raison est que la procédure requiert de stocker en mémoire la matrice Hessienne estimée dont la taille est égale au carré du nombre de variables à minimiser. A titre d'exemple, son utilisation pour entraîner un PMC de quelques milliers de poids n'est pas pratique à moins de disposer de ressources suffisantes en mémoire et en puissance de calcul.

Dans notre cas, nous avons au plus trois hyper-paramètres à optimiser pour le noyau KMOD, deux hyper-paramètres pour le noyau RBF, etc (en considérant la variable C). Le Hessien aura alors les dimensions (3x3) et (2x2) respectivement.

Du point de vue implantation, nous considérons l'espace logarithmique $\log\theta$ au lieu de θ. Dans le cas du RBF, nous effectuons l'optimisation dans l'espace $(\ln C, \ln\sigma^2)$. Dans le cas du KMOD, nous considérons l'espace $(\ln C, \ln\gamma^2, \ln\sigma^2)$. Ceci permet de contourner les contraintes de positivité sur C, γ et σ^2.

La formulation du gradient est alors légèrement changée. Ainsi, le vecteur de gradient de l'erreur E par rapport au logarithme de l'hyper-paramètre θ s'écrit :

$$\frac{\partial E}{\partial \ln\theta} = \theta\frac{\partial E}{\partial\theta}.$$

Nous considérons une variante du Quasi-Newton dite *«Broyden-Fletcher-Goldfarb-Shanno» (BFGS)* [80]. Elle est décrite comme suit :

Soit θ^t la valeur d'un hyper-paramètre particulier à l'itération t, et soit $E(\theta)$ la fonction objective à minimiser :

- Calculer la direction de descente $p = -H_t \nabla E(\theta^t)$.
- Trouver $\theta^{t+1} = \theta^t + \lambda p$ en utilisant une procédure de LSM (de l'anglais «Line Search Minimization») afin d'estimer un pas de descente adéquat.
- Réestimer la matrice H_{t+1} de l'itération suivante selon :

$$H_{t+1} = (I - \frac{sy^T}{y^T s}) H_t (I - \frac{ys^T}{y^T s}) + \frac{ss^T}{y^T s},$$

où $s = \theta^{t+1} - \theta^t$ et $y = \nabla E(\theta^{t+1}) - \nabla E(\theta^t)$.

Dans le l'algorithme ci-haut, H_t est une approximation de l'inverse du Hessien de la fonction objective. Par ailleurs, l'amplitude de l'incrément λ doit respecter l'inégalité :

$$E(\theta^t + \lambda p) \leq E(\theta^t) + \sigma_1 \lambda \nabla E(\theta^t)^T p,$$

où $0 < \sigma_1 < 1$ est une constante positive.

La direction de descente est estimée par $p = -H_t \nabla E(\theta^t)$ et requiert que H_t soit définipositif. L'algorithme du BFGS assure que H_{t+1} hérite de la positivité de H_t tant que

$$y^T s > 0.$$

Cette dernière est garantie si le Hessien initial est la matrice identité (son inverse H_0 l'est aussi) et que le pas de descente respecte la condition :

$$\nabla E(\theta^t + \lambda p) \geq \sigma_2 \nabla E(\theta^t)^T p,$$

où $0 < \sigma_1 < \sigma_2 < 1$.

Notons aussi que la direction de descente n'est pas garantie et que la procédure est très sensible au pas λ utilisé dans le LSM. D'où le risque d'instabilité numérique si λ est exagéré. Une solution consiste à contraindre le LSM à chercher dans une région prédéfinie.

Ci-dessous est décrit l'algorithme de minimisation de l'erreur empirique avec la méthode de Quasi-Newton :

Algorithme

1) Initialiser $\underline{\theta}$ à $\underline{\theta}_0$,

2) tant que pas convergence :

   a) apprentissage du SVM,

   b) estimer les paramètres A et B de la fonction sigmoïde,

   c) estimer la probabilité de l'erreur E,

   d) calculer le gradient de l'erreur $\frac{\partial}{\partial \underline{\theta}} E(\underline{\alpha}, \underline{\theta})$,

   e) estimer le Hessien H_t,

   f) modifier $\underline{\theta}$ selon : $\Delta \underline{\theta} = -\lambda H_t \frac{\partial}{\partial \underline{\theta}} E(\underline{\alpha}, \underline{\theta})$,

3) apprentissage du SVM avec $\underline{\theta}$ final.

L'algorithme LSM utilisé est une variante de la procédure *«Golden Section Minimization»* [80].

5.4 Minimisation de l'Erreur de Validation Croisée Généralisée (GACV)

Dans [122], Wahba et al. mettent en relief la connexion entre la formulation classique du SVM telle que introduite par Vapnik et la formulation du classifieur à marge molle [124] dont on en a introduit la définition dans un précédent chapitre. Ils démontrent que ce modèle équivaut au SVM L1 qui impose une pénalisation linéaire aux erreurs d'apprentissage.

Afin d'optimiser les hyper-paramètres, les auteurs proposent une borne supérieure du GCKL dite *«Generalized Comparative Kullback Lebleir»* qui est un estimé pessimiste de l'erreur de validation croisée. Cette erreur désignée *GACV* est définie *«Generalized Approximate Cross Validation»*. Dans ce qui suit, nous appellerons ce critère : Erreur de

validation croisée Généralisée. Il s'écrit :

$$GACV(\underline{\alpha},\underline{\theta}) = \frac{1}{l}\left(\sum_{i=1}^{l}\xi_i + 2\sum_{y_i f_i<-1}\alpha_i K_{ii} + \sum_{y_i f_i\in[-1,1]}\alpha_i K_{ii}\right); \tag{5.21}$$

où ξ_i représente la variable ressort associée à l'observation x_i et définie par :

$$\xi_i = 0 \quad si\ y_i f_i > 1$$
$$\xi_i = 1 - y_i f_i \quad si\ y_i f_i \geq 1,$$

où K_{ii} représente la valeur du noyau $K(x_i, x_i)$.

Ce critère exploite trois configurations différentes des vecteurs de support. La première catégorie que nous désignons S_1, représente tous les vecteurs de support situés sur la frontière de la marge du bon coté de la frontière de séparation. Les sorties brutes du SVM dans ce cas ont une valeur de ± 1. Les multiplicateurs α_i associés respectent l'inégalité $0 < \alpha_i < C$. Les vecteurs de support de la catégorie S_3 sont situés à l'intérieure de la marge. Leurs sorties appartiennent au domaine défini par $-1 \leq f_i \leq 1$. Parmi ces vecteurs de support, tous les exemples dont l'amplitude de ξ_i est supérieure à 1, représentent des erreurs d'apprentissage. Enfin S_2 représente la catégorie d'erreurs situées au delà de la marge. La figure 19 illustre les trois configurations sur un problème non séparable à deux dimensions.

D'après l'équation 5.21, le GACV pénalise deux fois plus les vecteurs de support de la catégorie S2 que ceux de la catégorie S1 et S3.

Pour un noyau RBF, $K_{ii} = 1, \forall i$, et l'expression du GACV devient :

$$GACV(\underline{\alpha},\underline{\theta}) = \frac{1}{l}\left(\sum_{i=1}^{l}\xi_i + 2\sum_{y_i f_i<-1}\alpha_i + \sum_{y_i f_i\in[-1,1]}\alpha_i\right).$$

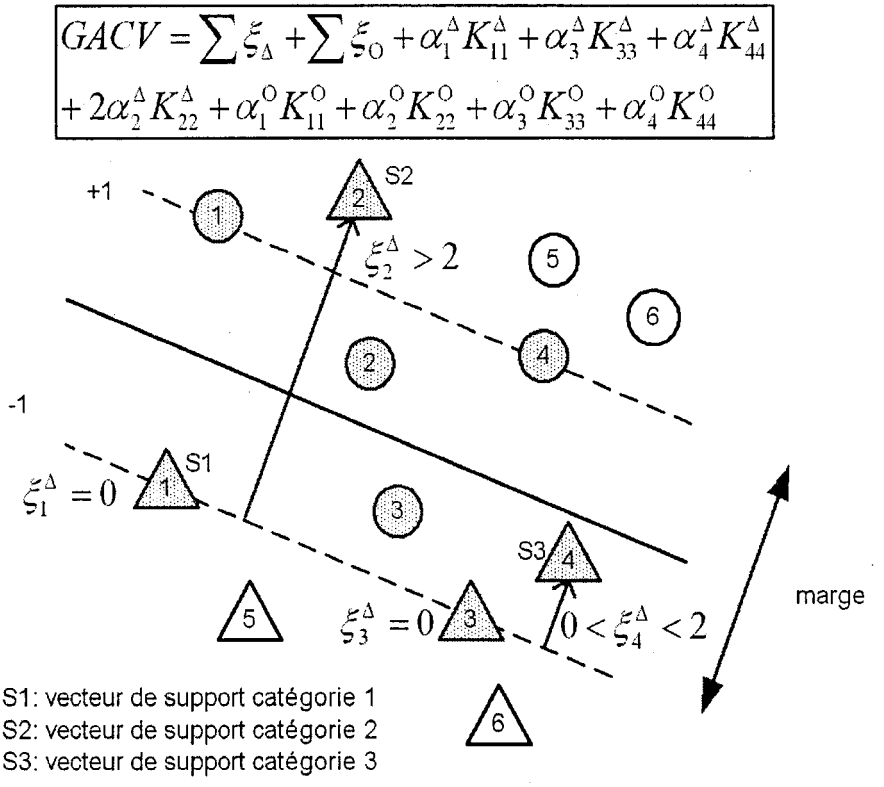


Figure 19 Deux classes non séparables et trois catégories de vecteurs de support

Dans le cas séparable, la minimisation du critère tente de réduire les valeurs de α_i. Ceci peut conduire à une diminution progressive du nombre de vecteurs de support. Nous constaterons cette propriété dans l'étude expérimentale que nous présentons dans un chapitre ultérieur.

5.4.1 Algorithme

L'Erreur de Validation Croisée Généralisée fournit une approximation du premier ordre d'une borne pessimiste de l'erreur de généralisation du SVM. Nous proposons dans ce qui suit un nouvel algorithme d'optimisation des hyper-paramètres du SVM basé sur la minimisation du GACV. Nous introduisons deux variantes de l'algorithme. La première

qu'on appellera GACVL1, est dédiée au modèle L1 du SVM. Sa forme générale s'écrit :

$$\frac{1}{l}\left(\sum_{i=1}^{k}\xi_i + c(i)\alpha_i\right).$$

La deuxième est dédiée au SVM L2. Elle est de la forme :

$$\frac{1}{l}\left(\sum_{i=1}^{k}\frac{1}{2}\xi_i^2 + c(i)\alpha_i\right).$$

La variable k représente le nombre de vecteurs de support. La variable $c(i)$ est une constante de régularisation qui dépend du vecteur de support associé. La différence entre les deux formulations réside dans la prise en compte de l'erreur empirique ξ_i.

Afin de minimiser les critères, nous utilisons une procédure de descente de gradient. Ci-dessous, nous décrivons l'algorithme de minimisation du GACVL1. Celui du GACVL2 est assez similaire au premier sauf pour le terme de l'erreur empirique.

Soit maintenant $\nabla GACV(\underline{\theta}) = \partial GACV(\underline{\alpha},\underline{\theta})/\partial\underline{\theta}$ le vecteur gradient du critère par rapport aux hyper-paramètres du classifieur. Ce vecteur est de dimension égale au nombre des hyper-paramètres n. Il s'écrit : $\frac{\partial}{\partial\underline{\theta}}GACV(\underline{\alpha},\underline{\theta}) =$

$$\frac{1}{l}\left(\sum_{i=1}^{l}\frac{\partial\xi_i(\theta)}{\partial\underline{\theta}} + 2\sum_{y_i f_i<-1}\left(K_{ii}\frac{\partial\alpha_i}{\partial\underline{\theta}} + \alpha_i\frac{\partial K_{ii}}{\partial\underline{\theta}}\right) + \sum_{y_i f_i\in[-1,1]}\left(K_{ii}\frac{\partial\alpha_i}{\partial\underline{\theta}} + \alpha_i\frac{\partial K_{ii}}{\partial\underline{\theta}}\right)\right).$$

Le gradient du critère peut s'écrire aussi : $\frac{\partial}{\partial\underline{\theta}}GACV(\underline{\alpha},\underline{\theta}) =$

$$\frac{1}{l}\left(\sum_{i=1}^{l}\frac{\partial\xi_i(\theta)}{\partial\underline{\theta}} + 2\sum_{i\in S_2}\left(K_{ii}\frac{\partial\alpha_i}{\partial\underline{\theta}} + \alpha_i\frac{\partial K_{ii}}{\partial\underline{\theta}}\right) + \sum_{i\in S_1\bigvee i\in S_3}\left(K_{ii}\frac{\partial\alpha_i}{\partial\underline{\theta}} + \alpha_i\frac{\partial K_{ii}}{\partial\underline{\theta}}\right)\right),$$

où le vecteur gradient du dépassement ξ_i par rapport aux hyper-paramètres $\underline{\theta}$ s'écrit :

$$\frac{\partial\xi_i(\underline{\theta})}{\partial\underline{\theta}} = \frac{\partial\xi_i(\theta)}{\partial f_i}\frac{\partial f_i}{\partial\underline{\alpha}}\frac{\partial\underline{\alpha}}{\partial\underline{\theta}} = \begin{cases} -y_i\frac{\partial f_i}{\partial\underline{\alpha}}\frac{\partial\underline{\alpha}}{\partial\underline{\theta}} \; si \; i \in S1, S2, S3 \\ \underline{0} \; sinon \end{cases} \tag{5.22}$$

Ci-dessus, $\underline{0}$ désigne le vecteur nul. Le calcul de l'expression de $\partial\underline{\alpha}/\partial\underline{\theta}$ est réalisé comme décrit dans l'équation 5.20. Reste alors à estimer les gradients $\partial K_{ii}/\partial\underline{\theta}$ qui dépendent du noyau utilisé. Ci dessous est présenté l'algorithme de minimisation du GACV tel que décrit plus haut.

Algorithme

1) Initialiser $\underline{\theta}$ à $\underline{\theta}_0$,

2) tant que pas convergence :

   a) apprentissage du SVM,

   b) calculer le gradient $\frac{\partial}{\partial\underline{\theta}}GACV(\underline{\alpha},\underline{\theta})$,

   c) modifier $\underline{\theta}$ selon : $\Delta\underline{\theta} = -\eta\frac{\partial}{\partial\underline{\theta}}GACV(\underline{\alpha},\underline{\theta})$,

3) apprentissage du SVM avec $\underline{\theta}$ final,

où η représente le pas d'apprentissage correspondant à l'itération courante. Sa valeur peut varier pendant l'optimisation si une procédure de LSM est utilisée.

Enfin, notons que le GACVL2 s'écrit :

$$GACV(\underline{\alpha},\underline{\theta}) = \frac{1}{l}\left(\sum_{i=1}^{l}\frac{\xi_i^2}{2} + 2\sum_{y_i f_i<-1}\alpha_i K_{ii} + \sum_{y_i f_i\in[-1,1]}\alpha_i K_{ii}\right). \tag{5.23}$$

L'estimé de son gradient diffère quelque peu de celle du GACVL1 à cause du terme $\xi_i^2/2$ dont la dérivée s'écrit : $\xi_i\frac{\partial\xi_i(\theta)}{\partial\underline{\theta}}$.

D'où l'expression du gradient du GACVL2 qui s'écrit :

$$\frac{1}{l}\left(\sum_{i=1}^{l}\xi_i\frac{\partial\xi_i(\theta)}{\partial\underline{\theta}} + 2\sum_{i\in S_2}\left(K_{ii}\frac{\partial\alpha_i}{\partial\underline{\theta}} + \alpha_i\frac{\partial K_{ii}}{\partial\underline{\theta}}\right) + \sum_{i\in S_1 \bigvee i\in S_3}\left(K_{ii}\frac{\partial\alpha_i}{\partial\underline{\theta}} + \alpha_i\frac{\partial K_{ii}}{\partial\underline{\theta}}\right)\right).$$

Dans la partie expérimentale nous tenterons de comparer la minimisation des deux critères sur un problème de données synthétiques que nous décrirons.

5.5 Conclusion

Trouver des valeurs adéquates des hyper-paramètres du SVM requiert de choisir un critère de sélection de modèle qui permet d'estimer l'erreur ou une borne supérieure de l'erreur. En outre, la sélection de modèle automatique nécessite une procédure d'optimisation qui minimise [4] le critère.

Dans ce chapitre, nous proposons une nouvelle méthodologie de sélection de modèle automatique du SVM basée sur l'estimation et la minimisation d'une erreur empirique. Cette dernière représente la probabilité d'erreur sur la partition des données de validation.

Nous avons aussi analysé deux autres critères de sélection de modèle basés respectivement sur la dimension VC et GACV. Le premier dérive des travaux de Vapnik sur le risque structurel. C'est une approximation pessimiste de l'erreur de généralisation. Le deuxième par ailleurs, est une approximation analytique de l'erreur de validation croisée LOO. Pour le dernier, nous avons en outre proposé un schéma de minimisation automatique pour les deux variantes L1 et L2 du SVM.

[4] Un critère de performance doit être maximisé par ailleurs.

CHAPITRE 6

OPTIMISATION DES HYPER-PARAMÈTRES DU SVM : CAS MULTICLASSE

6.1 Introduction

Nous avons abordé dans le chapitre 5 la problématique de l'optimisation des hyper-paramètres du SVM lors d'une classification binaire. Dans cette même partie, nous avons présenté trois algorithmes différents d'optimisation. Parmi ces schémas, deux sont des critères statistiques basés respectivement sur le principe du risque structurel (VC dim.) et sur une approximation de l'erreur de validation croisée (GACV). Nous avons aussi présenté une nouvelle méthodologie permettant l'optimisation des paramètres de noyaux basée sur la minimisation de l'erreur empirique estimée sur un ensemble indépendant des données.

En présence de données multiclasses, comme il est le cas en reconnaissance de chiffres, nous avons vu au chapitre 3 (section 3.11) qu'il est nécessaire de partager la tâche de classification entre plusieurs SVM et de combiner les réponses de chacun des classifieurs en vue de prédire l'appartenance d'une observation donnée. A la différence des classifieurs monolithiques tels que le perceptron multicouche ou le réseau RBF, considérer un ensemble de SVM pour traiter les données multiclasses rend plus complexe la tâche d'optimisation, d'autant plus que le nombre d'hyper-paramètres dépend non seulement du noyau, mais aussi des dichotomies associées aux classifieurs. Dans l'approche un-contre-tous par exemple, il faut considérer nK variables à optimiser tandis que dans une approche un-contre-un, il existe $nK(K-1)/2$ variables à optimiser (en considérant que le même type de noyau est utilisé avec chaque classifieur et que le nombre d'hyper-paramètres du noyau est égal à n et que le nombre de classes est égal à K).

Nous présentons dans ce qui suit, deux approches d'optimisation des hyper-paramètres d'un ensemble de SVM sur des données multiclasses. Les deux méthodes sont dévelop-

pées pour le cas d'une stratégie un-contre-un pour l'apprentissage. Mais l'étude peut être étendue à d'autres types de dichotomies des classes de données.

6.2 Optimisation locale des hyper-paramètres de SVM

Le noyau du SVM possède un ou plusieurs paramètres qui régissent la nature et la complexité de la frontière de décision. Ainsi, au gré de ses valeurs, il est possible d'obtenir un classifieur de capacité réduite, moyenne ou grande. La capacité optimale du classifieur dépend de la nature des données et du nombre de classes des données. Aussi, la marge de séparation produite par un SVM discriminant un couple de classes (i, j) est vraisemblablement supérieure à celle séparant la classe i du restant des classes. Donc, considérer uniquement des couples de classes de façon individuelle produit vraisemblablement des frontières de décision simples avec un nombre réduit de vecteurs de support. Dès lors que l'approche locale est tout a fait compatible avec la stratégie un-contre-un adoptée, il serait naturel d'envisager l'optimisation de l'ensemble des frontières séparant les $K(K-1)/2$ couples de classes.

L'algorithme de minimisation de l'erreur empirique proposée dans le chapitre 5 peut être utilisé pour optimiser chacun des SVM sur autant de partitions de données qu'il y a de couples de classes. Cette stratégie considère chaque couple de classes (i, j) indépendamment du restant des classes. Au final, chaque SVM aura ses propres valeurs d'hyper-paramètres qui dépendent seulement des données des deux classes considérées. Nous désignons cette approche *«optimisation locale»* des hyper-paramètres par opposition à *«l'optimisation globale»* que nous présenterons dans le prochain paragraphe.

Ci-dessous est présenté le Meta-Algorithme de l'optimisation locale de l'ensemble de SVM dans l'approche un-contre-un.

Meta-Algorithme

1) Pour i,j = 0 : K et i < j

2) Tant que pas convergence

   a) Apprentissage **classifieur(i,j)** sur **{apprentissage(i,j)}**

   b) Apprentissage **sigmoïde(i,j)** sur **{validation(i,j)}**

   c) Estimation **objective(i,j)** sur **{validation(i,j)}**

   d) Adaptation **hyper-paramètre(i,j)**

3) Pour i,j = 0 : K et i < j

4) Apprentissage **classifieur(i,j)** sur partition **{apprentissage(i,j)** $\bigcup$ **validation(i,j)}**

Dans l'algorithme décrit ci-haut, (i, j) permet d'indiquer le classifieur, les partitions de données d'apprentissage et de validation, la fonction d'activation (sigmoïde) permettant le calcul des probabilités à posteriori et les hyper-paramètres du classifieur considéré. L'algorithme présenté est celui utilisé sur les bases de données INDCENPARMI, USPS (voir chapitre 8). Étant donné qu'aucune de ces bases de données ne contient de données de validation, nous séparons l'ensemble d'apprentissage original en deux partitions : une partition de validation pour l'optimisation des hyper-paramètres et une partition d'apprentissage pour l'estimation du modèle. Une fois l'optimisation terminée, chaque classifieur est entraîné une dernière fois sur l'union des deux partitions composant l'ensemble original d'apprentissage. Ainsi donc, pour chaque couple de classes possible, nous entraînons le SVM sur les données correspondantes de la partition d'apprentissage. Ensuite, les données correspondantes de la partition de validation sont utilisées pour l'identification des paramètres de la sigmoïde (paramètres A et B). Ces derniers sont ensuite utilisés pour l'estimation de la probabilité d'erreur et l'estimation du gradient de la probabilité d'erreur (fonction objective). Enfin, après chaque correction des valeurs des hyper-paramètres, le SVM est ré-entraîné et ainsi de suite jusqu'à convergence de l'algorithme.

Ceci étant, on pourrait s'attendre à optimiser le comportement global de l'ensemble de classifieurs minimisant ainsi l'erreur multiclasse. Dans ce qui suit, nous présentons une méthode alternative d'optimisation dans l'approche un-contre-un qui prend en compte la totalité des classes disponibles en minimisant une estimation de l'erreur multiclasse.

6.3 Optimisation globale des hyper-paramètres de SVM

Cette section décrit une nouvelle méthode d'optimisation des hyper-paramètres de SVM dans l'approche un-contre-un. L'approche maximise la vraisemblance des données par minimisation de l'erreur quadratique sur l'ensemble de validation.

Soit D la matrice de décision associée à l'approche un-contre-un donnée par :

$$D = \begin{bmatrix} - & g_{01} & g_{02} & \cdots & g_{09} \\ g_{10} & - & g_{12} & \cdots & g_{19} \\ g_{20} & g_{21} & - & \cdots & g_{29} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{90} & g_{91} & g_{92} & \cdots & - \end{bmatrix} \tag{6.1}$$

où g_{ji} représente le vote du classifieur (j, i) étant donnée une observation de test x à classifier, tels que $(j, i)_{j,i=0,\ldots,K-1, j \neq i}$ et avec $g_{ji} = 1 - g_{ij}$ (K étant le nombre de classes).

En posant :

$$O_j = \frac{1}{K-1} \sum_{i, i \neq j}^{K-1} g_{ji}, \tag{6.2}$$

la règle de décision pourrait s'écrire :

$$c = \arg \max_{0 \leq j \leq K-1} O_j. \tag{6.3}$$

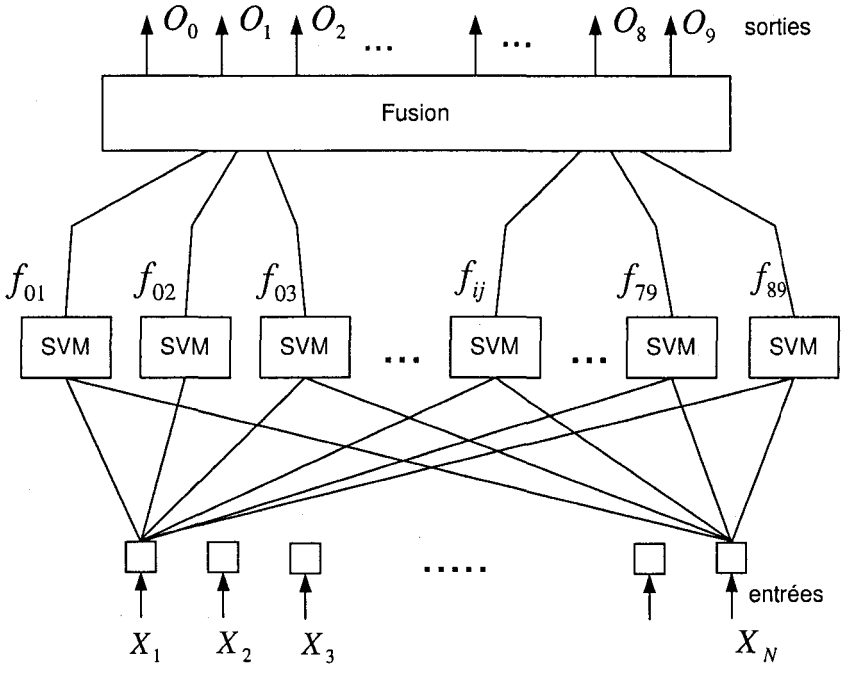


Figure 20 Architecture du système en stratégie un-contre-un pour K=10

En supposant que les variables g_{ji} représente la probabilité $P(x \in j | x \in i \cup j)$, O_j représente alors une estimation de la probabilité à posteriori $P(c = j|x)$ de la classe j sachant l'observation x.

Il est possible de maximiser la vraisemblance des données en minimisant la fonction de coût suivante :

$$C = \sum_{j=0}^{K-1} (O_j - y_j)^2, \tag{6.4}$$

où pour une observation x appartenant à la classe $c = k$,

$$y_j = \begin{cases} 1 & si\ j = k \\ 0 & sinon \end{cases}$$

représente l'étiquette de l'observation x associée à la sortie O_j (Figure 20).

Cette fonction de coût est une erreur quadratique dont la minimisation maximise la vraisemblance des données sous l'hypothèse que le bruit des étiquettes des données est de nature gaussienne. La fonction de coût C pourrait s'écrire encore :

$$C = \sum_{j=0}^{K-1} C_j = \sum_{j=0}^{K-1} ((\frac{1}{K-1} \sum_{i=0, i \neq j}^{K-1} g_{ji}) - y_j)^2. \tag{6.5}$$

où C_j représente l'erreur résiduelle associée à la sortie O_j et s'écrit :

$$C_j = (O_j - y_j)^2.$$

6.3.1 Minimisation de l'erreur multiclasse

Il est possible d'optimiser l'hyper-paramètre θ_{ij} associé au classifieur (i, j) dans l'approche un-contre-un de façon à tenir compte de l'erreur multiclasse définie par :

$$\frac{\sum_{i=1}^{N} [l(x_i) \neq c_i]}{N},$$

où N représente le nombre d'exemples x_i composant l'ensemble de données, $l(x_i)$ représente le vote issu de l'ensemble de SVM étant donné l'observation x_i, c_i représente l'étiquette de l'observation et $[z]$ est l'opérateur logique défini par :

$$[z] = \begin{cases} 1 & si\ z\ est\ vrai \\ 0 & sinon \end{cases}$$

Dans ce qui suit, nous décrivons une méthode de descente de gradient qui minimise le coût quadratique C par rapport à l'ensemble des hyper-paramètres θ_{ij} des classifieurs sur une partition des données indépendante des données d'apprentissage. Pour des raisons de consistance avec l'approche locale détaillée plus haut, nous désignerons cette partition des données ensemble de validation, et par exemples de validation l'ensemble des données qu'elle comprend.

Une descente de gradient est envisageable sur C sous réserve que les variables g_{ij} soient dérivables (leurs dérivées existent). Ne pouvant pas utiliser directement les sorties brutes f_{ij} des classifieurs comme des mesures de vote, nous adoptons en revanche une fonction d'activation de la même forme que celle utilisée dans le chapitre 5 dont nous rappelons l'équation qui s'écrit :

$$g(f) = \frac{1}{1 + \exp(Af + B)}$$

Soit donc à optimiser le paramètre θ_{ji} associé au classifieur (j, i). On pourrait écrire le gradient de l'erreur résiduelle C_j par rapport à θ_{ji} comme suit :

$$\frac{\partial C_j}{\partial \theta_{ji}} = \frac{\partial C_j}{\partial O_j} . \frac{\partial O_j}{\partial \theta_{ji}}.$$

Ainsi, le gradient :

$$\frac{\partial C_j}{\partial O_j} = 2.(O_j - y_j),$$

et le gradient :

$$\frac{\partial O_j}{\partial \theta_{ji}} = \frac{1}{K-1} . \frac{\partial g_{ji}}{\partial \theta_{ji}}.$$

L'estimé du gradient du coût globale C par rapport à ce paramètre est égal :

$$\frac{\partial C}{\partial \theta_{ji}} = \frac{\partial}{\partial \theta_{ji}}[(O_j - y_j)^2 + (O_i - y_i)^2], \tag{6.6}$$

ce qui donne :

$$\frac{\partial C}{\partial \theta_{ji}} = 2.\frac{\partial O_j}{\partial \theta_{ji}}(O_j - y_j) + 2.\frac{\partial O_i}{\partial \theta_{ji}}(O_i - y_i),$$

qu'on peut transcrire en :

$$\frac{\partial C}{\partial \theta_{ji}} = \frac{2}{K-1}(O_j - y_j)\frac{\partial g_{ji}}{\partial \theta_{ji}} + \frac{2}{K-1}(O_i - y_i)\frac{\partial g_{ij}}{\partial \theta_{ji}}.$$

Or,

$$g_{ij} = 1 - g_{ji},$$

d'où

$$\frac{\partial g_{ij}}{\partial \theta_{ji}} = -\frac{\partial g_{ji}}{\partial \theta_{ji}}.$$

Enfin,

$$\frac{\partial C}{\partial \theta_{ji}} = \frac{2}{K-1}\frac{\partial g_{ji}}{\partial \theta_{ji}}((O_j - y_j) - (O_i - y_i)), \tag{6.7}$$

avec

$$O_j = \frac{1}{K-1}\sum_{k=0,k\neq j}^{K-1} g_{jk}$$

et

$$O_i = \frac{1}{K-1}\sum_{k=0,k\neq i}^{K} g_{ik}.$$

Soit $\frac{\partial C(x)}{\partial \theta_{ji}}$ l'estimé du gradient du coût global C par rapport à θ_{ji} étant donnée l'observation x appartenant à la classe k.

A partir de l'équation 6.7, nous pouvons dériver trois formes possibles de l'expression du gradient du coût global C étant donné un hyper-paramètre θ_{ji} à optimiser et une observation x de l'ensemble de validation. On trouve donc :

$$\frac{\partial C(x)}{\partial \theta_{ji}} = \begin{cases} \frac{2}{K-1}\frac{\partial g_{ji}(x)}{\partial \theta_{ji}}(O_j(x) - O_i(x) - 1) & \texttt{si j=k} \\ \frac{2}{K-1}\frac{\partial g_{ji}(x)}{\partial \theta_{ji}}(1 + O_j(x) - O_i(x)) & \texttt{si i=k} \\ \frac{2}{K-1}\frac{\partial g_{ji}(x)}{\partial \theta_{ji}}(O_j(x) - O_i(x)) & \texttt{sinon} \end{cases} \tag{6.8}$$

Cette équation met en relief la différence fondamentale entre l'approche locale d'optimisation et l'approche globale. En effet, l'expression du gradient du coût global C par rapport à un hyper-paramètre θ_{ji} donné, prend en compte non seulement les exemples des classes i et j, mais aussi le restant des exemples de l'ensemble de validation et qui n'appartiennent pas au couple de classes (i,j). Les solutions f_{ij} de l'approche globale se retrouvent modifiées à cause de l'effet des exemples des classes marginales. Ces derniers tendent à minimiser la différence $O_j(x) - O_i(x)$.

L'estimation du gradient du coût global est ensuite considérée sur la totalité des observations de l'ensemble de validation selon l'équation :

$$\frac{\partial \bar{C}}{\partial \theta_{ji}} = \frac{1}{N} \sum_{x_k, k=1}^{N} \frac{\partial C(x_k)}{\partial \theta_{ji}}. \tag{6.9}$$

Par ailleurs, de même que dans le chapitre 5, le gradient du vote g_{ji} par rapport à θ_{ji} peut être développé selon :

$$\frac{\partial g_{ji}}{\partial \theta_{ji}} = \frac{\partial g_{ji}}{\partial f_{ji}} \frac{\partial f_{ji}}{\partial \theta_{ji}}.$$

Les expressions des gradients $\frac{\partial g}{\partial f}$ et $\frac{\partial f}{\partial \theta}$ sont développées dans les l'équation 5.18, 5.19 et 5.20 du même chapitre. Au final, ayant pu estimer les dérivées partielles $\frac{\partial g_{ji}}{\partial \theta_{ji}}$ pour chaque observation de l'ensemble de validation, la détermination de l'amplitude de la correction $\triangle\ \theta_{ji}$ à appliquer à θ_{ji} est égale à :

$$\triangle\ \theta_{ji} = -\eta_{glob} \frac{\partial \bar{C}}{\partial \theta_{ji}} = -\eta_{glob} \frac{1}{N} \sum_{x_k, k=1}^{N} \frac{\partial C(x_k)}{\partial \theta_{ji}}. \tag{6.10}$$

Durant tout le développement, nous avons considéré un seul hyper-paramètre θ_{ji} associé au classifieur (j,i). Cette hypothèse simplifie les notations et ne restreint en rien la capacité de l'algorithme à optimiser plus d'un hyper-paramètre par classifieur. En effet, on pourrait remplacer le scalaire θ_{ji} par le vecteur $\underline{\theta_{ji}}$ et utiliser la règle d'apprentissage suivante

$$\triangle\ \underline{\theta_{ji}} = -\eta_{glob} \frac{\partial \bar{C}}{\partial \underline{\theta_{ji}}} = -\eta_{glob} \frac{1}{N} \sum_{x_k, k=1}^{N} \frac{\partial C(x_k)}{\partial \underline{\theta_{ji}}}, \tag{6.11}$$

où les quantités $\frac{\partial \bar{C}}{\partial \underline{\theta_{ji}}}$ et $\frac{\partial C(x_k)}{\partial \underline{\theta_{ji}}}$ sont des vecteurs de dimension égale au nombre d'hyper-paramètres associées au classifieur (j, i).

Ci-dessous est présenté le meta-algorithme d'optimisation globale de l'ensemble de SVM dans l'approche un-contre-un.

Meta-Algorithme

1) Tant que pas convergence
   a) Pour i,j = 0 : K-1 et i < j
      i) Apprentissage **classifieur(i,j)** sur **{apprentissage(i,j)}**
      ii) Apprentissage **sigmoïde(i,j)** sur **{validation(i,j)}**
   b) Estimation **objective** sur **{validation}**
   c) Pour i,j = 0 : K-1 et i < j
      i) Adaptation **hyper-paramètres(i,j)**
2) Pour i,j = 0 : K-1 et i < j
3) Apprentissage **classifieur(i,j)** sur partition **{apprentissage(i,j)** $\bigcup$ **validation(i,j)}**

Comme dans l'approche locale d'optimisation, nous partageons l'ensemble d'apprentissage original en deux partitions. Une partition d'apprentissage pour l'entraînement des classifieurs et une partition de validation pour leur optimisation. Sur cette dernière sont estimées les fonctions d'activation (sigmoïde) nécessaires à l'estimation des probabilités à posteriori. Après quoi, les probabilités conditionnelles des classes sont estimées pour chacune des observations de la partition de validation. Enfin, la fonction de coût (fonction objective) ainsi que les gradients du coût par rapport à chacun des hyper-paramètres sont estimés. Les hyper-paramètres sont alors corrigés, et de nouveau les classifieurs sont entraînés. L'algorithme continue ainsi jusqu'à sa convergence.

En comparaison avec l'optimisation locale des hyper-paramètres, l'optimisation globale présente deux particularités importantes :

- les classifieurs sont entraînés sur la partition d'apprentissage en utilisant uniquement le couple de classes courant, mais sont optimisés en utilisant l'ensemble des données de validation disponibles (toutes les classes comprises).

- les modèles sont optimisés simultanément à chaque itération de l'algorithme. Il est alors nécessaire de garder l'ensemble des classifieurs en mémoire.

6.4 Conclusion

La classification de données multiclasses requiert de construire et de combiner plusieurs SVM selon un schéma de décomposition particulier. L'approche un-contre-un permet de construire un SVM pour chaque couple de classes disponible.

Dans ce chapitre, nous avons présenté deux méthodologies de sélection de modèle pour un ensemble de SVM dans l'approche un-contre-un. La première, dite 'approche locale', permet d'optimiser les paramètres du noyau en considérant seulement les données du couple de classes à séparer. Les frontières de décision sont alors optimisées localement. Dans la deuxième, nous avons proposé un critère d'optimisation globale des SVM de l'ensemble en maximisant la vraisemblance des données de validation. Celle-ci est l'erreur quadratique entre les probabilités à posteriori estimées des classes et les probabilités désirées des données.

CHAPITRE 7

EXPÉRIMENTATION : SÉLECTION DE MODÈLES POUR DES DONNÉES BICLASSES

Nous présentons dans cette partie du travail les résultats d'expérience des méthodes proposées et discutées dans le chapitre 5. Nous y trouverons une description exhaustive des protocoles d'expérience adoptés ainsi qu'une analyse des résultats obtenus. Cette étude s'articule autour de trois parties.

Dans la première, nous considérons un problème simple à deux classes que nous testons avec les méthodes de la dimension VC et GACV. Dans la deuxième, nous considérons un problème synthétique biclasse plus complexe constitué de deux classes bi-modales. Sur ce problème, nous comparerons les performances de l'erreur empirique avec les critères VC dim. et GACV. Pour ce dernier, nous considérons les deux variantes du critère, à savoir SVM L1 et SVM L2. Dans la troisième partie du travail, nous étudions l'effet de la taille de la base de validation sur la performance de l'optimisation par réduction de l'erreur empirique.

7.1 Introduction

Nous avons présenté dans le chapitre 5 trois méthodes d'optimisation des hyper-paramètres qui utilisent les critères suivants :

1. la dimension VC,
2. l'erreur de validation croisée généralisée,
3. l'erreur empirique.

Dans ce qui suit, nous allons appliquer ces trois méthodes d'optimisation à un problème de classification binaire non séparable. Plus loin, dans le chapitre 8, le cas multiclasse

sera sujet à une expérimentation plus approfondie sur la base de données chiffres indiens INDCENPARMI et USPS [6].

Toute au long des expérience, nous nous intéressons à :

- la minimisation du critère de sélection de modèle,
- l'erreur après convergence,
- la complexité du classifieur après convergence.

7.2 Données synthétiques biclasses

Nous avons produit deux problèmes binaires synthétiques de natures différentes représentant des données biclasse. Dans le premier ensemble, chacune des classes est représentée par une distribution gaussienne normale et sphérique d'exemples dans un espace à deux dimensions. Les classes sont équiprobables. Leurs probabilités conditionnelles s'écrivent :

$$p(x|y=1) = \frac{1}{2}\mathcal{N}\left(\begin{pmatrix} +1 \\ +1 \end{pmatrix}, \begin{pmatrix} 0.7 & 0 \\ 0 & 0.7 \end{pmatrix}\right)$$

et

$$p(x|y=-1) = \frac{1}{2}\mathcal{N}\left(\begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} 0.7 & 0 \\ 0 & 0.7 \end{pmatrix}\right)$$

Ici, $\mathcal{N}(\mu, \Sigma)$ représente une distribution gaussienne de vecteur de moyenne μ et de matrice de covariance Σ.

Nous avons produit un ensemble de données de cette distribution contenant 120 exemples à raison de 60 données par classe (Figure 21).

Le deuxième problème que nous considérons représente deux classes non-linéairement séparables. Leurs distributions sont bimodales et symétriques par rapport à l'origine disposées en *OU EXCLUSIF* (Figure 22). La solution optimale à ce problème est fortement non-linéaire et assez proche d'une hyperbole. Notons aussi que les deux classes sont équiprobables.

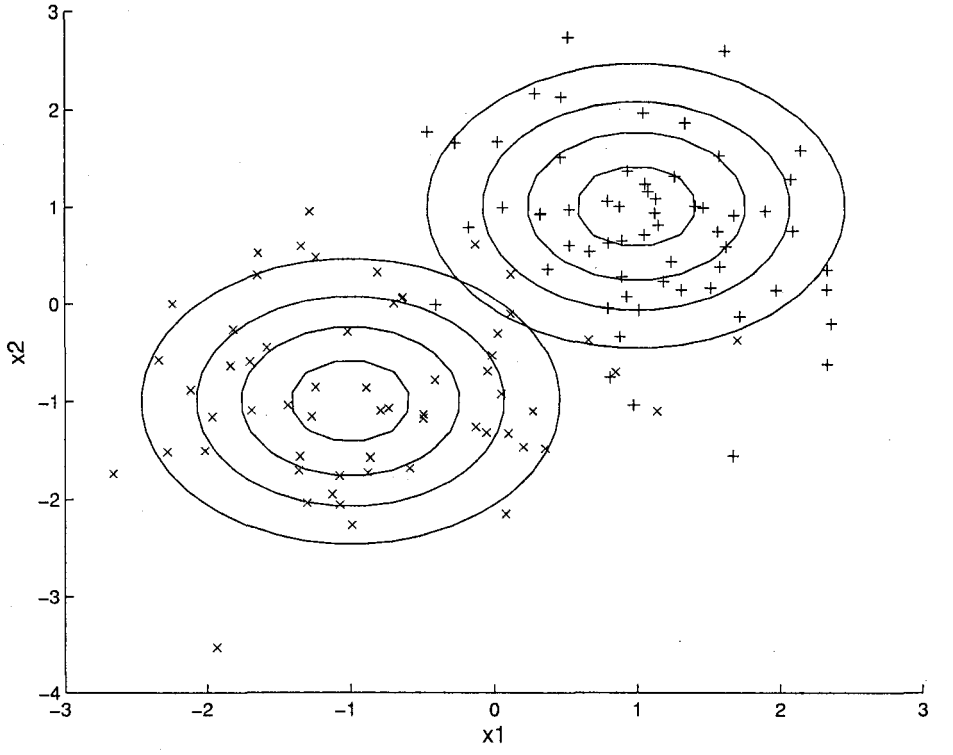


Figure 21 Données synthétiques biclasses représentant le problème *GAUSS2*

La probabilité d'observer un exemple de la classe positive s'écrit

$$p(x|y=1) = \frac{1}{2}\mathcal{N}\left(\begin{pmatrix} +2 \\ +2 \end{pmatrix}, \begin{pmatrix} 0.9 & 0 \\ 0 & 0.9 \end{pmatrix}\right) + \frac{1}{2}\mathcal{N}\left(\begin{pmatrix} -2 \\ -2 \end{pmatrix}, \begin{pmatrix} 0.9 & 0 \\ 0 & 0.9 \end{pmatrix}\right)$$

et la probabilité d'observer un exemple de la classe négative est

$$p(x|y=-1) = \frac{1}{2}\mathcal{N}\left(\begin{pmatrix} -2 \\ +2 \end{pmatrix}, \begin{pmatrix} 0.9 & 0 \\ 0 & 0.9 \end{pmatrix}\right) + \frac{1}{2}\mathcal{N}\left(\begin{pmatrix} +2 \\ -2 \end{pmatrix}, \begin{pmatrix} 0.9 & 0 \\ 0 & 0.9 \end{pmatrix}\right).$$

Nous avons généré trois ensembles indépendants issus de cette même distribution représentant respectivement les données d'apprentissage, les données de validation et les données de test. Chaque ensemble comprend 240 exemples.

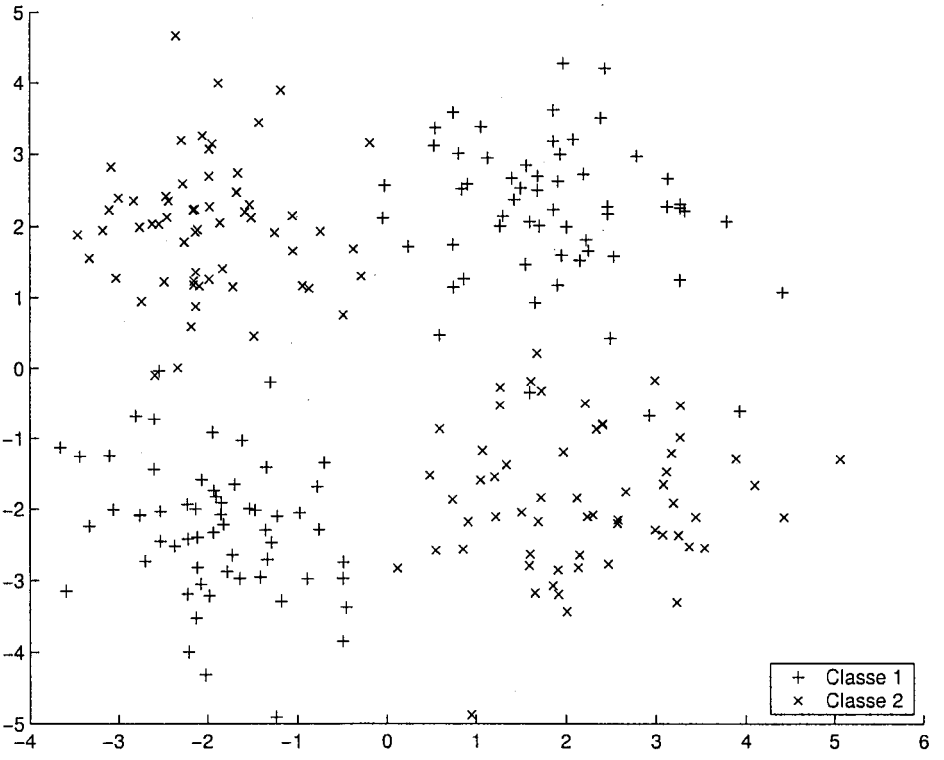


Figure 22 Données d'apprentissage du problème XOR

Dans ce qui suit, nous nous proposons d'évaluer les solutions produites par les différents critères d'optimisation. En particulier, nous analysons la précision de chacun d'eux ainsi que la compacité de la solution pour différentes configurations initiales.

7.2.1 Minimisation de la dimension VC

La dimension VC [1] d'un classifieur traduit à la fois son pouvoir discriminant et la compacité des données dans l'espace augmenté F (voir chapitre 3). C'est un critère analytique qui est estimé sur les données d'apprentissage seulement. Pour cette fin, nous n'avons donc pas besoin de recourir à un ensemble de données supplémentaire. Ceci constitue un avantage indéniable spécialement pour les applications ou une quantité suffisante des données n'est pas facile à obtenir.

[1] Le critère rayon-marge est une borne conservatrice de la dimension VC. Mais par abus de langage, on confondra les deux.

Nous avons vu que la minimisation de la dimension VC est réalisée en maximisant la marge séparant les classes et en minimisant l'hyper-sphère englobant les données. Ceci est traduit par le critère $\frac{R^2}{M^2}$ à minimiser pour R étant le rayon de l'hyper-sphère englobante et M étant la marge de séparation. Ce critère est valable lorsque les erreurs d'apprentissage du classifieur sont pénalisées suivant un coût quadratique. Ce modèle du SVM est aussi appelé SVM $L2$ à cause du type de pénalisation imposé aux points à l'intérieur et au delà de la marge. Dans ce cas ci, les multiplicateurs α_i ne sont plus contraints par la limite supérieure C du SVM $L1$. Néanmoins, dans le cas non séparable, une constante de régularisation est introduite pour assurer un compromis entre la minimisation de l'erreur d'apprentissage et la maximisation de la marge. Cette constante est équivalente au paramètre C qu'on connaît. Elle sert à mieux conditionner la matrice de Grahm Schmidt du SVM, en additionnant un terme d'amplitude inversement proportionnelle à C aux composantes diagonales de cette matrice.

Comme déjà expliqué, nous minimisons le critère avec une procédure de descente de gradient sur les paramètres du noyau et éventuellement le paramètre de compromis C. La méthode estime à chaque itération, la direction et l'amplitude de la correction à apporter en calculant la dérivée partielle de la dimension VC par rapport aux paramètres à optimiser. Cette estimation nécessite le calcul du gradient du carré du rayon R^2 et le gradient de l'inverse du carré de la marge $\frac{1}{M^2}$.

Étant donné que $\frac{1}{M^2}$ est égal à $\|w\|^2$, l'estimation de son gradient requiert seulement le modèle du SVM préalablement entraîné. D'autre part, le gradient de R^2 nécessite l'estimation de R^2 ainsi que les points de support associés à l'hyper-sphère. Son calcul requiert la résolution d'une optimisation quadratique avec contraintes dont la complexité est approximativement de même ordre que le calcul du modèle du SVM. Toutefois, lorsque les classes sont difficiles à séparer et pour un grand C, on constate que le temps de convergence vers une solution est plus lent pour le SVM que pour le calcul de l'hyper-sphère

Tableau VIII

Temps de calcul du SVM et de l'hyper-sphère en secondes pour plusieurs noyaux (dim=2 et $C = 10e4$)

Noyau	rbf	kmod	linéaire	sigmoïde
SVM	2.1	2.2	1.6	4.1
Hyper-sphère	2.0	1.3	1.7	3.2

7.2.1.1 Calcul de l'hyper-sphère

Le tableau VIII compare les durées d'apprentissage du SVM et de l'hyper-sphère pour plusieurs noyaux. L'algorithme d'optimisation quadratique utilisé est celui fourni dans la boite d'outils d'optimisation de MATLAB. Les tests ont été réalisés sur le problème GAUSS2 décrit plus haut.

Le tableau montre que les durées de calcul estimées varient selon type de noyau utilisé ainsi que les valeurs de ses paramètres. Dans le cas d'un noyau linéaire (pas de noyau), l'hyper-sphère est décrite par un minimum de deux points symétriques par rapport à son centre et situés sur son périmètre. Sinon, trois points sont nécessaires au moins. En règle générale, $n + 1$ exemples de support sont nécessaires pour définir une hyper-sphère dans un espace à n dimensions.

Notons, par ailleurs, que les deux durées sont approximativement de même ordre de grandeur. Le tableau IX montre les temps d'apprentissage du SVM et d'estimation de l'hyper-sphère pour différentes tailles des données. On remarque que l'estimation de l'hyper-sphère est de même ordre que la durée d'apprentissage pour une taille de 60, tandis qu'elle est approximativement le double pour les autres tailles. La raison en est que le terme linéaire de la fonction objective de l'hyper-sphère intègre un terme de noyau qui n'y est pas dans la fonction objective du SVM. Pour les noyaux RBF et KMOD, ses valeurs peuvent être remplacées par 1, accélérant ainsi son estimation. En général, ceci est uniquement le

Tableau IX

Temps de calcul du SVM et de l'hyper-sphère en secondes pour plusieurs tailles de données (noyau RBF avec $\sigma = 1$, dim=2 et $C = 10e4$)

Nombre d'exemples par classe	60	150	300
SVM	2.1	29.9	299.3
Hyper-sphère	2.0	47.5	512.4

Tableau X

Temps de calcul du SVM et de l'hyper-sphère en secondes pour plusieurs dimensions de données (noyau RBF avec $\sigma = 1$ et $C = 10e4$)

Dimension des données	2	10	100
SVM	2.1	1.1	1.3
Hyper-sphère	2.0	0.7	1.0

cas des noyaux sphériques dont la valeur $K(x, y)$ est fonction de la distance euclidienne $\|x - y\|$ entre les points de l'espace d'entrée x et y .

Dans le tableau X, nous montrons les durées d'apprentissage du SVM et d'estimation de l'hyper-sphère pour différentes dimensions des données. La taille des données est fixée à 60 exemples par classe. Ainsi, nous pouvons constater que le calcul de l'hyper-sphère est légèrement plus rapide que le calcul du SVM pour les hautes dimensions [2]. Pour les dimensions 10 et 100, les exemples sont plus clairsemés dans l'espace, et moins de vecteurs de support sont trouvés. Ceci a pour effet d'accélérer le calcul. Toutefois, le calcul des noyaux est proportionnel à la dimensionalité des données. Ce qui ralentit le calcul dans les deux cas. D'où les temps de calcul qui augmentent pour une dimensionalité de 100 dans le tableau.

[2] par haute dimension, nous entendons faible rapport entre la taille des données et leur dimension

7.2.2 Minimisation du GACV

Dans la partie abordant notre méthodologie, nous avons présenté l'Erreur de Validation Croisée Généralisée comme étant un critère analytique qui approxime l'erreur de validation croisée LOO.

Nous avons appliqué le critère avec ses variantes GACV L1 et GACV L2 sur les deux problèmes *GAUSS2* et *XOR*.

Dans le premier problème nous comparons le critère avec la dimension VC sur l'ensemble des données disponibles où nous estimons l'erreur de généralisation par une procédure de validation croisée sur dix partitions des données. Dans le problème *XOR*, nous comparons le critère avec la dimension VC et l'erreur empirique.

7.2.3 Minimisation de l'erreur empirique

Dans la section 7.3.2, nous présentons une étude comparative des résultats d'expériences du critère de l'erreur empirique que nous confrontons aux critères de la dimension VC et de l'Erreur de Validation Croisée Généralisée. Cette comparaison est réalisée pour différentes valeurs initiales de C et de σ. Dans un deuxième temps, nous évaluons l'efficacité de l'erreur empirique à estimer l'erreur de généralisation sur le problème XOR en utilisant des bases de validation de tailles différentes mais de même distribution. Pendant toute l'expérience, nous considérons le critère de l'erreur empirique avec le modèle SVM L1 qui ne permet pas l'optimisation simultanée du paramètre de régularisation C. Pour ce faire, il faut utiliser le modèle SVM L2.

7.3 Étude expérimentale

La descente de gradient est une méthode de minimisation sans contraintes qui se comporte bien lorsque la fonction objective est continue et dérivable. Il est démontré que la dimension VC est un critère continu, et qui peut être utilisé pour optimiser les paramètres du

noyau [24]. Nous avons implanté cette procédure telle décrite dans le chapitre 5. Notons toutefois quelques détails spécifiques concernant la manipulation.

Pour les noyaux KMOD et RBF, nous optimisons σ^2 à la place de $sigma$ dans l'espace des réels positifs. Ceci est une contrainte rendue possible en corrigeant les hyper-paramètres dans l'échelle logarithmique. Par exemple, la correction à apporter au paramètre x de façon à ce que $\ln x = y$, est égale à $\triangle\ x = e^{\triangle y}$ avec

$$\frac{\partial}{\partial x} = \frac{1}{x}.\frac{\partial}{\partial y}.$$

Quelque soit la valeur de y dans $\mathbb{R}$, x reste positif.

7.3.1 Problème GAUSS2

Sur ce problème, nous évaluons les procédures de minimisation de la dimension VC et de l'Erreur de Validation Croisée Généralisée. L'objectif de l'expérimentation est d'évaluer les algorithmes de minimisation des deux critères sur ce problème simple avant d'entreprendre une étude comparative de l'efficacité des critères avec celui de l'erreur empirique que nous proposons.

Nous utilisons un modèle SVM L2 pour la dimension VC, un SVM L1 pour le critère GACV L1 et un SVM L2 pour le GACV L2. Nous avons choisi d'utiliser un noyau RBF avec une valeur initiale de σ égale à 3.16. La valeur de C dans cette manipulation est fixée à 1000.

La figure 23 montre la configuration initiale et la frontière de décision du classifieur avant l'optimisation du noyau. Nous avons fixé le nombre d'itérations maximal de la descente de gradient à 200. Aucune procédure d'accélération n'a été appliquée au départ. L'algorithme est arrêté si l'amplitude du vecteur de gradient est inférieure à une valeur de tolérance

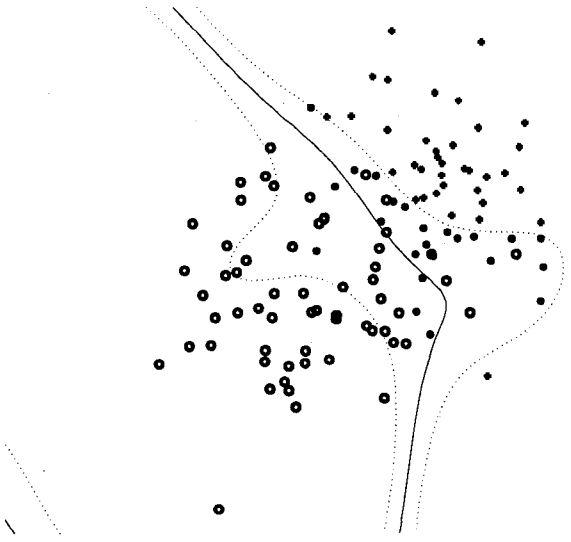


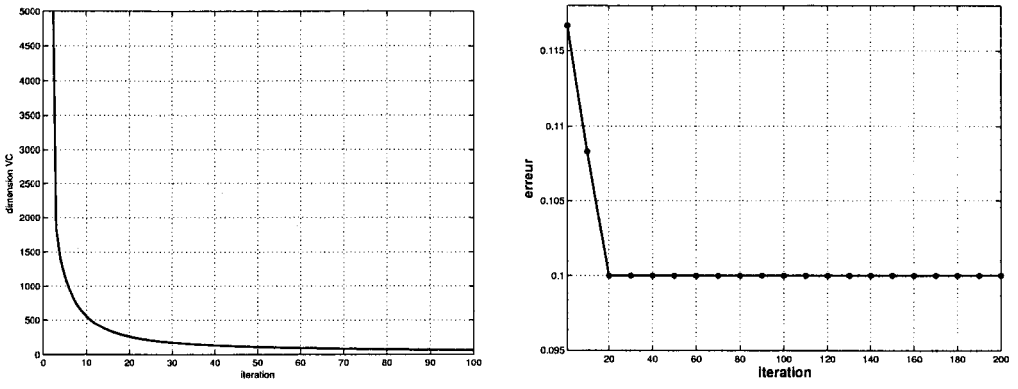
Figure 23 Solution initiale pour le problème synthétique *GAUSS2*

égale à 10^{-6}. Dans le cas contraire, la procédure est arrêté lorsque le nombre d'itérations maximum est atteint.

La figure 24 montre les courbes de variation de la dimension VC et de l'erreur de validation croisée pendant l'optimisation. L'erreur de validation croisée est estimée sur la base de dix partitions des données. Les deux courbes montrent une réduction de la dimension VC de 9649 à 37. La valeur du paramètre σ final est 131.2. Cette valeur du paramètre correspond à un noyau très large.

La figure 25 montre les frontières de décision du SVM après minimisation de la dimension VC. Notons que la solution est une quasi-droite assez proche du classifieur optimale de ce problème. L'erreur de test est réduite de 11.6% à 10%.

Par ailleurs, sur ce même problème, la minimisation du GACV L2 ne réussit pas à optimiser la largeur du noyau RBF. Les courbes de la figure 26 montrent l'évolution de la fonction objective et de l'erreur de test pendant la procédure d'optimisation. La figure montre que la réduction du GACV L2 (de 240 à 25) correspond à une augmentation de l'erreur de 11.6% à 17.5%. Nous pouvons constater d'autre part que la frontière de décision après



(a) Variation de la dimension VC (b) Variation de l'erreur «LOO»

Figure 24 Optimisation par minimisation du VC

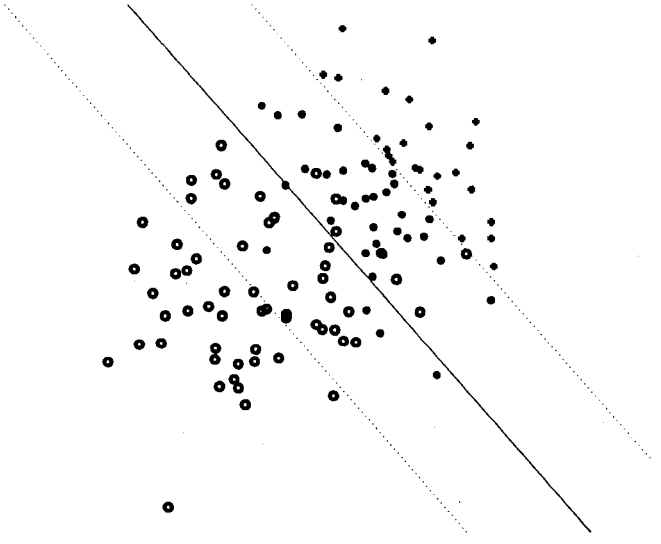
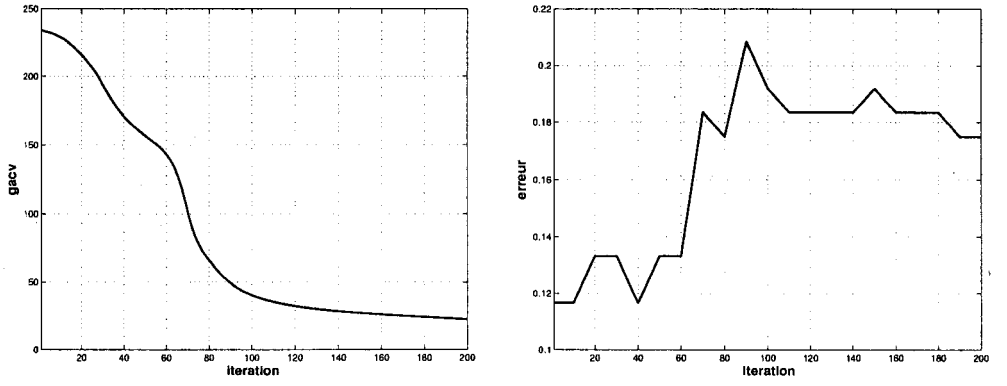


Figure 25 Solution finale avec la minimisation de la dimension VC sur le problème *GAUSS2*

optimisation est différente du classifieur optimal de ce problème (Figure 27).

L'expérience sur le problème *GAUSS2* a permis de valider notre algorithme de minimisation du GACV et de le comparer avec la procédure de minimisation de la dimension VC [24]. Il nous a été possible de constater que le GACV, dans les conditions de l'expérience en question, ne garantit pas la réduction de l'erreur. En effet, la performance des critères



(a) Variation du GACV (b) Variation de l'erreur «LOO»

Figure 26 Courbes montrant les variation du GACV et de l'erreur de validation croisée «LOO» pendant l'optimisation sur le problème GAUSS2

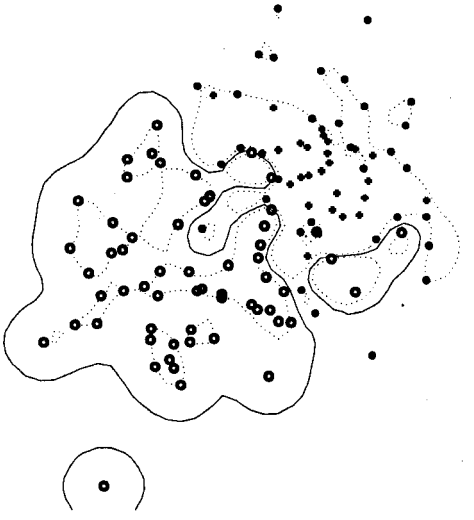


Figure 27 Solution finale de la minimisation du GACV sur le problème GAUSS2

d'optimisation étudiées dépend de beaucoup d'hypothèses sur la nature des données, leur taille et la complexité du problème.

7.3.2 Problème XOR

Dans le but de présenter une étude plus complète, nous appliquons les deux critères précédents sur le problème *XOR*. Aussi, une comparaison avec la minimisation de l'erreur empirique est réalisée dans les mêmes conditions d'expérience.

L'apprentissage du SVM L1 utilise la décomposition successive de SVM^{light} détaillée dans le chapitre 3 [46]. Afin d'inverser la matrice H dans l'équation 5.20, nous utilisons une décomposition LU. Cette procédure a une complexité en temps de l'ordre $O(n^3)$, n étant la dimension de la matrice H. Pendant l'optimisation, quelques problèmes numériques peuvent surgir si la matrice H est mal conditionnée. Ceci a lieu si quelques unes de ses valeurs propres sont assez proches de zéro. Afin de corriger cet inconvénient, nous ajoutons une constante de régularisation ϵ de l'ordre de 1e-7 aux éléments de la diagonale. Une autre technique possible est d'utiliser une variante de la méthode de Cholesky adaptée aux matrices semi-défini positives [126].

Nous décidons de choisir trois configurations initiales différentes. Le premier cas présente une configuration initiale du noyau RBF proche d'un minimum local avec une faible erreur d'apprentissage et un nombre réduit de vecteurs de support ($\sigma_0 = \sqrt{2}$). Dans le deuxième cas, nous choisissons une configuration de sur-apprentissage avec beaucoup de vecteurs de support et une très faible erreur d'apprentissage ($\sigma_0 = 0.1$). Enfin, dans la troisième configuration on tente de réduire le nombre initial de vecteurs de support en choisissant σ assez grand ($\sigma_0 = 45$). Notons qu'une très large valeur associée à σ réduit la complexité du classifieur à celle d'une équation affine dans l'espace d'entrée.

Pour chaque configuration initiale du noyau, nous utilisons trois valeurs différentes de C : 10000, 100 et 1. Pendant l'optimisation, nous enregistrons les valeurs de la fonction objective, du nombre de vecteurs de support et de l'erreur de test.

Tableau XI

Comparaison des résultats d'optimisation du noyau RBF avec différents critères sur le problème du XOR ($\sigma_0 = \sqrt{2}$)

		Erreur emp.	**Dimension VC**	**GACV L1**	**GACV L2**
c=10000	Obi	0.26	16000	199.1	143.2
	SVi	32	34	32	34
	Tsi (%)	9.58	10.0	9.58	10.0
	Obf	0.36	96.0	1.7	2.8
	SVf	97	225	124	132
	Tsf (%)	9.17	6.25	9.17	8.33
c=100	Obi	0.21	524.0	4.8	4.7
	SVi	40	48	40	48
	Tsi (%)	5.42	5.0	5.42	5.0
	Obf	0.21	110.0	2.4	4.5
	SVf	47	238	82	50
	Tsf (%)	5.42	34.6	5.38	5.42
c=1	Obi	0.19	16.3	0.25	0.3
	SVi	61	138	61	138
	Tsi (%)	3.75	4.17	3.75	4.17
	Obf	0.19	13	0.25	0.3
	SVf	51	240	62	138
	Tsf (%)	3.75	48.8	3.75	4.17

Le tableau XI compare les résultats de simulation des critères de l'erreur empirique avec la dimension VC, GACV L1 et GACV L2 pour la configuration initiale avec $\sigma_0 = \sqrt{2}$. Les symboles Obi (Obf), SVi (SVf), Tsi (Tsf) désignent respectivement les valeurs de la fonction objective, du nombre de vecteur de support et de l'erreur de test avant optimisation (après optimisation). Notons aussi que le critère de l'erreur empirique est estimé sur un ensemble de validation de 240 exemples.

Dans le tableau XI et pour C=10000, les modèles L1 et L2 du SVM ont presque la même performance avec des erreurs de test de 9.58% et 10% et un nombre de vecteurs de support

de 32 et 34 respectivement. Au final, l'ensemble des critères excepté l'erreur empirique sont réduits. La dimension VC dans ce cas est le critère qui réduit le plus l'erreur de test de 10% à 6.25%. Les erreurs de test dans le cas du GACV L1 et GACV L2 sont aussi réduites à 9.17% à 8.33%. Notons qu'en comparaison avec la configuration initiale, l'ensemble des critères augmente le nombre de vecteurs de support. Toutefois, l'erreur empirique donne le moins de vecteurs de support. Pour $C = 100$ et $C = 1$, l'erreur de test reste inchangée pour les critères de l'erreur empirique, GACV L1 et GACV L2 alors qu'elle est gravement détériorée pour la dimension VC. Pour les deux cas, la solution de l'erreur empirique est la plus compacte avec une cinquantaine de vecteurs de support.

Tableau XII

Comparaison des résultats d'optimisation du noyau RBF avec différents critères sur le problème du XOR ($\sigma_0 = 0.1$)

		Erreur emp.	**Dimension VC**	**GACV L1**	**GACV L2**
	Obi	0.41	109.37	0.92	0.92
	SVi	238	238	238	238
c=10000	Tsi (%)	27.08	27.08	27.08	27.08
	Obf	0.36	114.24	0.92	0.92
	SVf	98	240	238	238
	Tsf (%)	9.17	39.58	27.08	27.08
	Obi	0.41	108.33	0.92	0.92
	SVi	238	238	238	238
c=100	Tsi (%)	27.08	27.5	27.08	27.5
	Obf	0.21	112.83	0.92	0.92
	SVf	46	240	238	238
	Tsf (%)	5.42	38.75	27.08	27.5
	Obi	0.41	56.36	0.91	1.06
	SVi	238	240	238	240
c=1	Tsi (%)	28.33	27.92	28.33	27.92
	Obf	0.19	56.22	0.91	1.06
	SVf	51	240	238	240
	Tsf (%)	3.75	27.92	28.33	27.92

Dans le tableau XII, nous rapportons les résultats d'optimisation lorsqu'un sur-apprentissage initial prononcé des données y est présent. Pour les trois valeurs de C, le critère empirique fournit le classifieur le plus précis et le plus compacte. Le nombre de vecteurs de support est réduit jusqu'à 20% du nombre initial pour $C = 100$. Les erreurs de test sont égales à celles trouvées lors de la procédure initiale, et sont 9.17% ($C = 10000$), 5.42% ($C = 100$) et 3.75% ($C = 1$).

Par ailleurs, la procédure d'optimisation semble converger au même paramètre initial du RBF pour les critères GACV L1 et GACV L2 où la fonction objective reste à 0.92 pour $C = 10000$ et $C = 100$. Ces valeurs sont d'ailleurs inférieures à celles du tableau XI pour les mêmes valeurs de C. Ceci n'améliore pas l'erreur de test pour autant. Pour $C = 1$, la fonction objective reste à 0.92 et 1.06 pour GACV L1 et GACV L2 respectivement. Dans ces deux cas, la procédure de descente de gradient semble s'immobiliser sur un plateau supérieur. Dans le tableau XI, les deux critères sont réduits avec la même valeur de C.

Concernant la dimension VC, notons que la procédure d'optimisation ne réduit pas le critère et détériore l'erreur de test pour $C = 10000$ et $C = 100$. La fonction objective n'est probablement pas convexe dans ce cas. Pour $C = 1$, l'algorithme converge à une valeur proche de la dimension VC initiale mais ne réduit pas l'erreur.

Le tableau XIII montre les résultats d'optimisation pour $\sigma_0 = 45$. La dimension VC, bien que réduite pour les trois valeurs de C, détériore l'erreur de test.

Seul GACV L2 réduit l'erreur pour $C = 10000$ à 4.17%. La minimisation de GACV L1 accentue l'erreur par ailleurs. L'optimisation de l'erreur empirique ne converge pas pour l'ensemble de validation utilisé (240 exemples). Le tableau XVI montre par ailleurs que l'erreur est réduite à 5.0% pour $C = 10000$, bien que la fonction objective ne soit pas convexe dans cette région du paramètre σ (vu que l'erreur empirique augmente à la fin de l'optimisation). D'ailleurs, il semble que celle ci ne soit pas convexe pour une très grande valeur de C ($C = 10000$ dans notre cas) et un faible nombre de vecteurs

Tableau XIII

Comparaison des résultats d'optimisation du noyau RBF avec différents critères sur le problème du XOR ($\sigma_0 = 45$)

		Erreur emp.	Dimension VC	GACV L1	GACV L2
	Obi		5786	2301.7	1839.1
	SVi	N.D	91	57	91
c=10000	Tsi (%)		5.0	6.67	5.0
	Obf		102	1.0	689.0
	SVf	N.D	240	240	29
	Tsf (%)		52.08	50.0	4.17
	Obi	0.31	236.3	98.14	76.13
	SVi	235	240	235	240
c=100	Tsi (%)	25.0	5.83	25.0	5.83
	Obf	0.22	12.6	31.32	75.03
	SVf	55	240	77	240
	Tsf (%)	5.83	50.83	7.5	5.42
	Obi	0.5	3.13		2.48
	SVi	240	240	N.D	240
c=1	Tsi (%)	50.0	42.5		42.5
	Obf	0.5	2.55		2.48
	SVf	240	240	N.D	240
	Tsf (%)	50.0	43.33		42.5

de support. Pour $C = 100$, l'erreur empirique réduit l'erreur de test de 25.0% à 5.83%, GACV L1 de 25.0% à 7.5% et GACV L2 de 5.83% 5.42%. Le nombre de vecteurs de support est respectivement 55, 77 et 240 pour l'erreur empirique, GACV L1 et GACV L2. Pour $C = 1$, les critères de l'erreur empirique, la dimension VC et GACV L2 s'arrêtent à la première itération où la descente de gradient semble s'arrêter à un point stationnaire.

Toutefois, le tableau XVI montre que l'erreur empirique s'arrête à un minimum de 3.75% d'erreur et une cinquantaine de vecteurs de support pour des ensembles de validation de taille 480 et 720. La figures 28 et 29 montrent les frontières de décision du SVM L1 avant

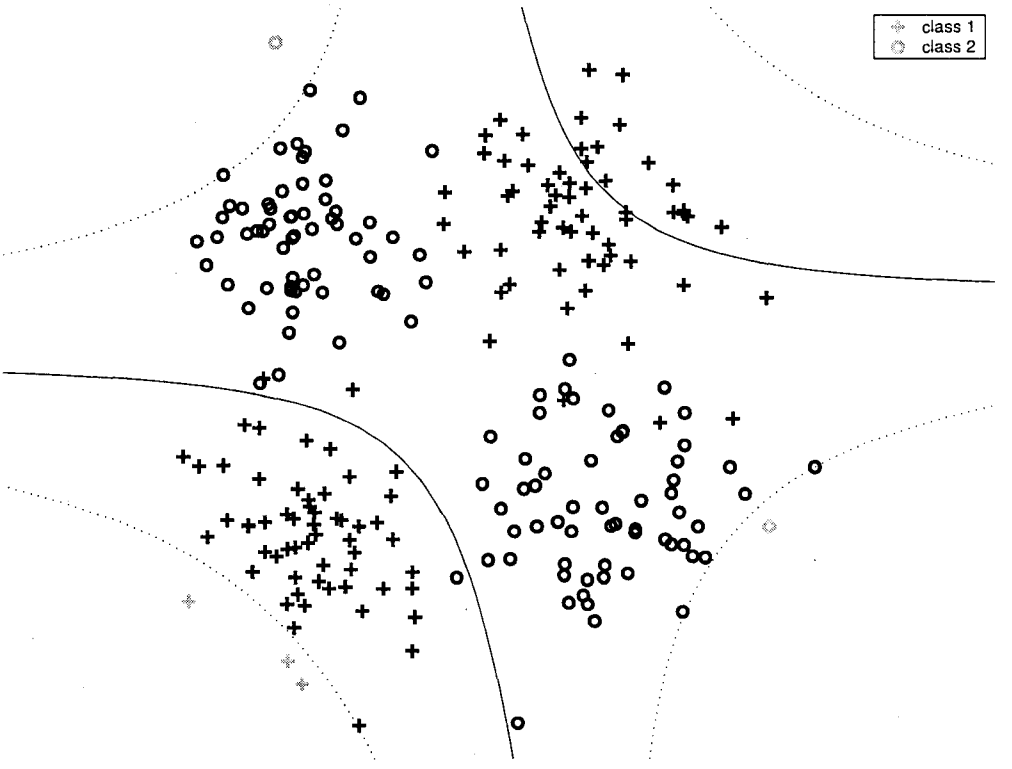


Figure 28 Frontière de décision initiales du SVM L1 avant optimisation avec le critère GACV L1 pour $C = 100$ et $\sigma_0 = 45$. Les points noirs représentent des vecteurs de support

et après minimisation du critère GACV L1 lorsque C=100. La figure 30 montre la frontière de décision du SVM L2 après minimisation du critère GACV L2 lorsque C=100.

7.3.3 Influence de la taille de l'ensemble de validation

Les tableaux XIV, XV et XVI présentent les résultats d'expérience de la procédure de minimisation de l'erreur empirique pour différentes tailles de l'ensemble de validation.

Le tableau XIV compare les résultats d'optimisation avec des ensemble de validation de taille 240, 480 et 720. Nous avons fait en sorte de respecter le même protocole que précédemment en considérant trois valeurs différentes de C. Les résultats rapportés sont pour $\sigma_0 = \sqrt{2}$. Il est facile de constater que pour cette configuration initiale la taille de l'ensemble de validation n'améliore pas l'erreur de test pour les trois valeurs de C considérées.

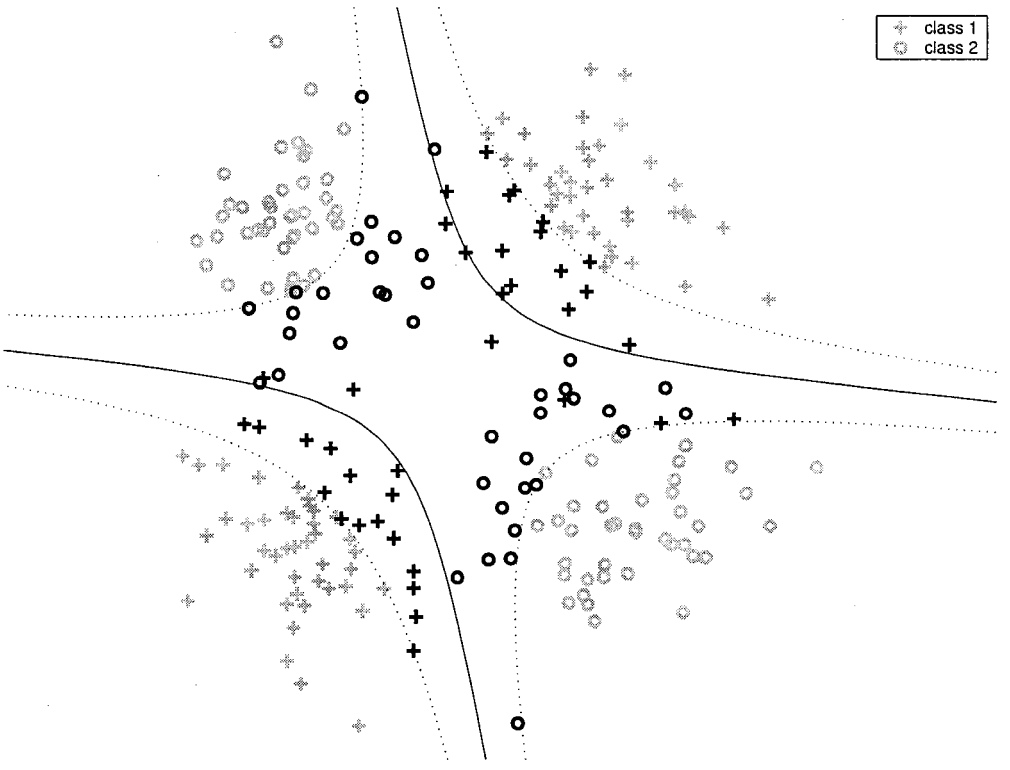


Figure 29 Frontière de décision du SVM L1 obtenue après optimisation avec le critère GACV L1 pour $C = 100$ et $\sigma_0 = 45$. Les points noirs représentent des vecteurs de support

Néanmoins, le nombre de vecteurs de support est légèrement moindre pour l'ensemble de validation à 720 exemples excepté pour $C = 1$, où la taille de l'ensemble de validation ne semble pas influencer le nombre de vecteurs de support. Ceci est compréhensible dans la mesure où une faible valeur de C réduit assez la complexité du classifieur pour que l'effet des paramètres soit notablement affaibli. Notons par ailleurs, qu'il est difficile d'observer une quelconque réduction de l'erreur lorsqu'on sait que la solution initiale (avant optimisation) est assez proche d'un minimum local comme l'est notre cas.

Le tableau XV confirme la réduction de vecteurs de support pour une taille 720 à des valeurs de C égales à 10000 et 100. Les erreurs de test sont restées tout de même insensibles à la taille de l'ensemble de validation. Dans le tableau XVI, nous montrons les résultats d'expérience avec trois tailles différentes de l'ensemble de validation pour une valeur ini-

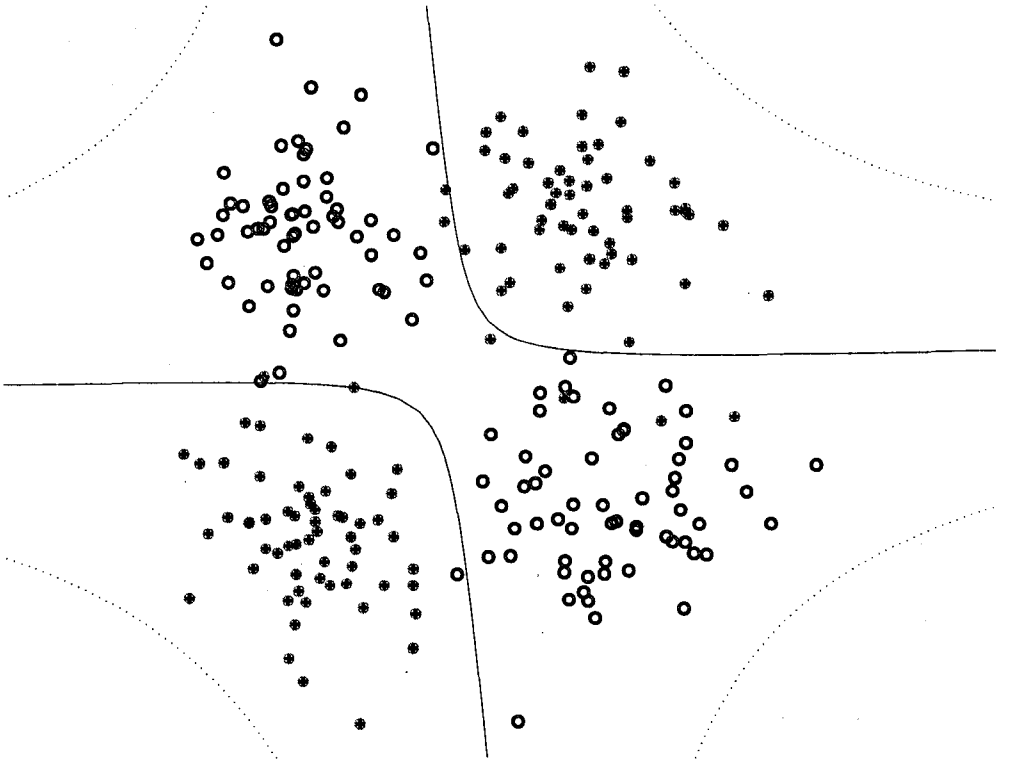


Figure 30 Frontière de décision du SVM L2 obtenue après optimisation avec le critère GACV L2 pour $C = 100$ et $\sigma_0 = 45$

tiale de σ égale à 45. Nous constatons que pour $C = 10000$, l'erreur de test est réduite de 6.67% à 5.0% pour 720 exemples de validation alors qu'elle augmente de 6.67% à 9.17% pour 480 exemples de validation considérés. Notons aussi la réduction notable du nombre de vecteurs de support de 57 à 23 pour le premier. Pour $C = 100$, nous remarquons une légère réduction de l'erreur et du nombre de vecteurs de support pour les tailles 480 et 720 en comparaison avec 240. Enfin, pour $C = 1$, la procédure d'optimisation produit une solution plus que satisfaisante pour les ensembles à 480 et 720 exemples (une erreur de test égale à 3.75% et une cinquantaine de vecteurs de support). Pour 240 exemples, le gradient de l'erreur à la première itération est très faible et la procédure s'arrête.

Tableau XIV

Comparaison des résultats d'optimisation du noyau RBF avec l'erreur empirique pour différentes tailles de l'ensemble de validation sur le problème du XOR ($\sigma_0 = \sqrt{2}$)

Taille		240	480	720
	Obi	0.26	0.24	0.23
	SVi	32	32	32
c=10000	Tsi (%)	9.58	9.58	9.58
	Obf	0.36	0.35	0.37
	SVf	97	99	92
	Tsf (%)	9.17	9.17	9.17
	Obi	0.21	0.20	0.20
	SVi	40	40	40
c=100	Tsi (%)	5.42	5.42	5.42
	Obf	0.21	0.20	0.20
	SVf	47	43	41
	Tsf (%)	5.42	5.42	5.42
	Obi	0.19	0.17	0.17
	SVi	61	61	61
c=1	Tsi (%)	3.75	3.75	3.75
	Obf	0.19	0.18	0.17
	SVf	51	50	51
	Tsf (%)	3.75	3.75	3.75

Les figures 31, 32 et 33 présentent respectivement les courbes de variation de la fonction objective, du nombre de vecteurs de support et de l'erreur de test pour les trois critères lorsque $C = 100$.

7.4 Conclusion

La minimisation de l'erreur empirique a été comparée à la minimisation du VC et GACV sur un problème de données biclasses. L'erreur empirique est plus performante et plus stable. Elle réduit efficacement la complexité du SVM en élaguant le maximum de vecteurs de support inutiles. Par ailleurs, GACV et VC ne permettent pas d'améliorer l'erreur

Tableau XV

Comparaison des résultats d'optimisation du noyau RBF avec l'erreur empirique pour différentes tailles de l'ensemble de validation sur le problème du XOR ($\sigma_0 = 0.1$)

Taille		240	480	720
	Obi	0.41	0.41	0.41
	SVi	238	238	238
c=10000	Tsi (%)	27.08	27.08	27.08
	Obf	0.36	0.35	0.37
	SVf	98	99	93
	Tsf (%)	9.17	9.17	9.17
	Obi	0.41	0.41	0.41
	SVi	238	238	238
c=100	Tsi (%)	27.08	27.08	27.08
	Obf	0.21	0.20	0.20
	SVf	46	46	40
	Tsf (%)	5.42	5.42	5.42
	Obi	0.41	0.41	0.41
	SVi	238	238	238
c=1	Tsi (%)	28.33	28.33	28.33
	Obf	0.19	0.19	0.18
	SVf	51	50	50
	Tsf (%)	3.75	3.75	3.75

lorsque le classifieur sur-apprend les données. En outre, la dimension VC n'est pas toujours une fonction objective convexe. Aussi, le SVM L2 fournit une solution plus complexe que le SVM L1 et produit un nombre plus important de vecteurs de support. Dans l'erreur empirique, agrandir l'ensemble de validation permet de réduire davantage l'erreur avec moins de vecteurs de support.

Tableau XVI

Comparaison des résultats d'optimisation du noyau RBF avec l'erreur empirique pour différentes tailles de l'ensemble de validation sur le problème du XOR ($\sigma_0 = 45$)

Taille		240	480	720
	Obi		0.25	0.25
	SVi	N.D	57	57
c=10000	Tsi (%)		6.67	6.67
	Obf		0.35	0.30
	SVf	N.D	99	23
	Tsf (%)		9.17	5.0
	Obi	0.31	0.31	0.30
	SVi	235	235	235
c=100	Tsi (%)	25.0	25.0	25.0
	Obf	0.22	0.21	0.20
	SVf	55	42	46
	Tsf (%)	5.83	5.42	5.42
	Obi	0.50	0.50	0.50
	SVi	240	240	240
c=1	Tsi (%)	50.0	50.0	50.0
	Obf	0.50	0.19	0.18
	SVf	240	50	51
	Tsf (%)	50.0	3.75	3.75

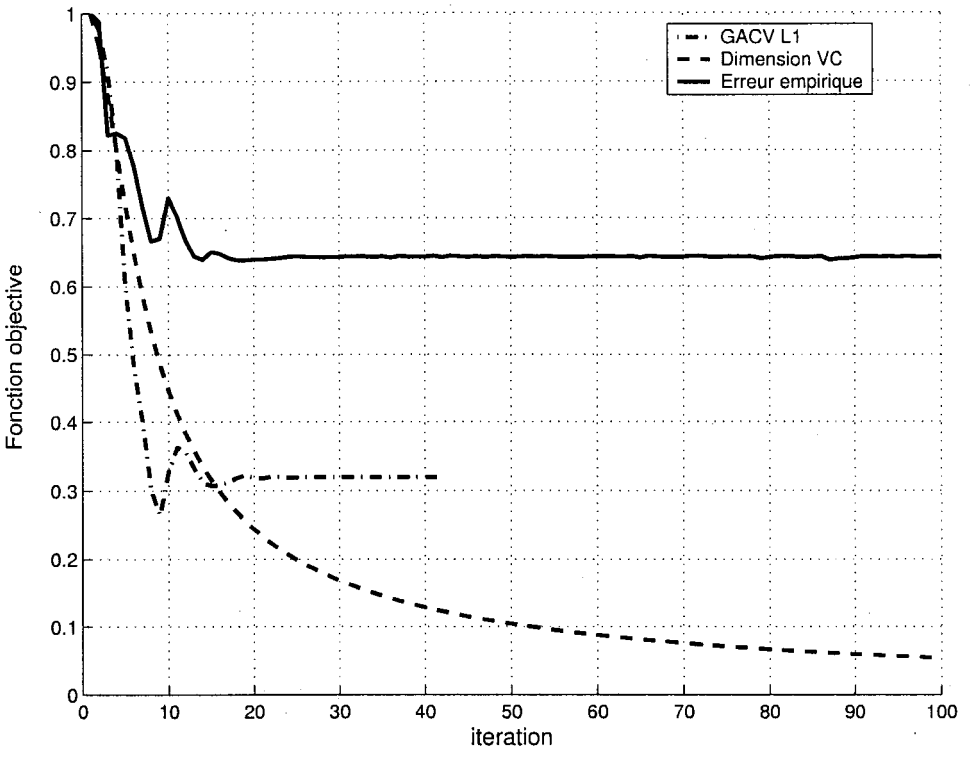


Figure 31 Variation de la fonction objective au cours de l'optimisation pour $C = 100$ et $\sigma_0 = 45$

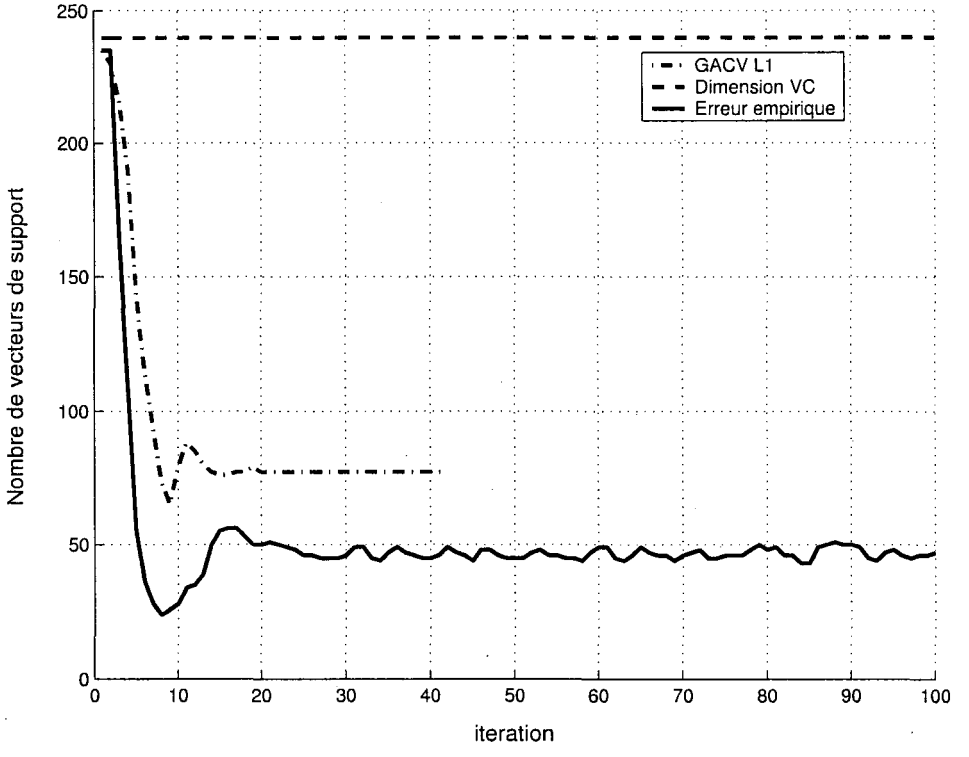


Figure 32 Variation du nombre de vecteurs de support au cours de l'optimisation pour $C = 100$ et $\sigma_0 = 45$

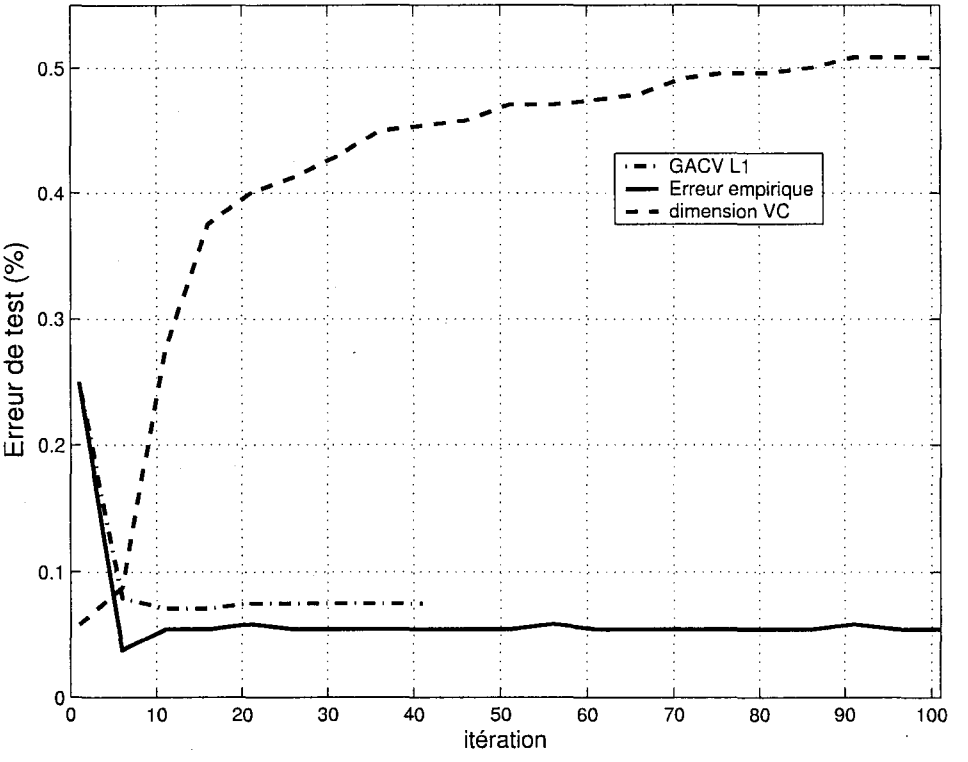


Figure 33 Variation de l'erreur de test au cours de l'optimisation pour $C = 100$ et $\sigma_0 = 45$

CHAPITRE 8

EXPÉRIMENTATION : OPTIMISATION DE SVMS SUR DES DONNÉES MULTICLASSES

Nous présentons dans ce chapitre la partie expérimentale validant le critère de l'erreur empirique sur des données synthétiques et des données réelles. Dans un premier temps nous démontrons la justesse de l'optimisation globale et la comparons avec l'optimisation locale. Dans un deuxième temps nous présentons les résultats d'expérience sur des images de chiffres manuscrits non contraints.

Dans la section 8.1, nous présentons des résultats d'expérience de l'optimisation globale obtenus sur un ensemble de données synthétiques représentant un problème à trois classes et deux attributs.

Dans la section 8.2, nous démontrons l'efficacité de notre méthodologie sur deux bases de données d'images. La première est USPS représentant des images de chiffres arabes et autour de laquelle plusieurs travaux font référence. La deuxième est la base de données de chiffres Indiens de CENPARMI pour laquelle nous avons élaboré un ensemble de primitives dédiées qu'on expliquera.

8.1 Données synthétiques multiclasse

Dans le chapitre 6 (section 6.3), nous avons développé une nouvelle méthodologie de sélection de modèle pour la sélection des paramètres de noyaux dans l'approche Un-contre-Un. Cette méthode cherche à minimiser l'erreur multiclasse en estimant les probabilités à posteriori $P(w_i|x), \forall i = 0, \ldots, K-1$ (K étant le nombre de classes) et en minimisant l'erreur quadratique :

$$E = \sum_{i=0}^{K-1} (P(w_i|x) - y_i)^2, \tag{8.1}$$

où y_i représente l'étiquette binaire (0 ou 1) de l'observation x par rapport à la sortie i.

A présent, nous présentons les résultats d'expériences de l'approche globale d'optimisation sur un problème synthétique à trois classes. Ce dernier est constitué des données du problème du XOR déjà présenté dans le chapitre 7 plus une troisième classe représentée par une gaussienne centrée autour de l'origine. Nous avons généré trois ensembles de données indépendants et de tailles égales représentant les données d'apprentissage, de validation de test. L'ensemble d'apprentissage sert à entraîner les trois classifieurs discriminant les couples de classes $(\bullet, \times)$, $(o, \times)$ et $(\bullet, o)$. Les trois classes sont distribuées équitablement sur les trois ensembles à hauteur de 240 exemples par classe. La figure 34 représente les données d'apprentissage du problème étudié. Dans la figure, les points des classes $\bullet, \times$ et o sont affichés en bleu, rouge et vert respectivement.

Lors de cette expérience, nous avons utilisé une descente de gradient simple sans aucune procédure d'accélération. La fonction de coût minimisée est l'erreur quadratique multiclasse de l'ensemble de SVMs comme décrit dans le chapitre 6. Nous avons fixé un nombre maximal d'itérations égal è 200.

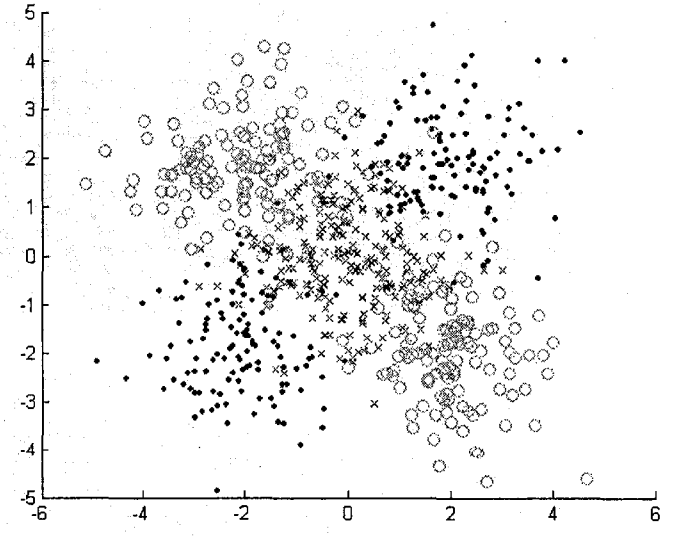


Figure 34 Données d'apprentissage du problème à trois classes

Nous avons considéré le problème d'optimisation des trois paramètres de noyaux RBF $\sigma_{(\bullet,\times)}$, $\sigma_{(o,\times)}$ et $\sigma_{(\bullet,o)}$ associés aux trois couples de classes $(\bullet, \times)$, $(o, \times)$ et $(\bullet, o)$. Nous avons utilisé une valeur de C égale à 100.

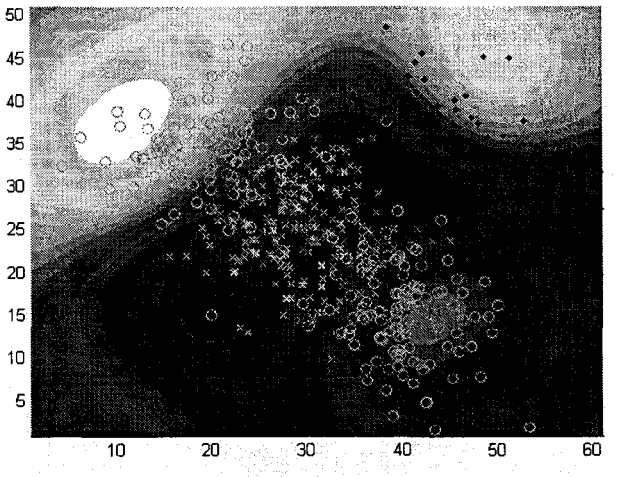
La figure 35 montre les frontières de décision finales obtenues après optimisation avec l'approche d'optimisation locale (courbe (a)) et l'approche d'optimisation globale (courbe (b)). Les couleurs dans la figure représentent des probabilités d'appartenance à chacune des classes o et $\times$.

La courbe (a) de la figure 36, montre l'évolution de la fonction objective optimisée (fonction de coût) le long de l'optimisation globale. Sa valeur représente l'erreur quadratique entre le vecteur d'étiquettes désirées et le vecteur de sortie de l'ensemble de SVMs. Sa valeur converge à la quatre-vingtième itération. Nous pouvons constater que la minimisation de cette valeur s'accompagne d'une minimisation des erreurs de validation des trois classifieurs comme montré par les courbes (b), (c) et (d) de la figure 36.

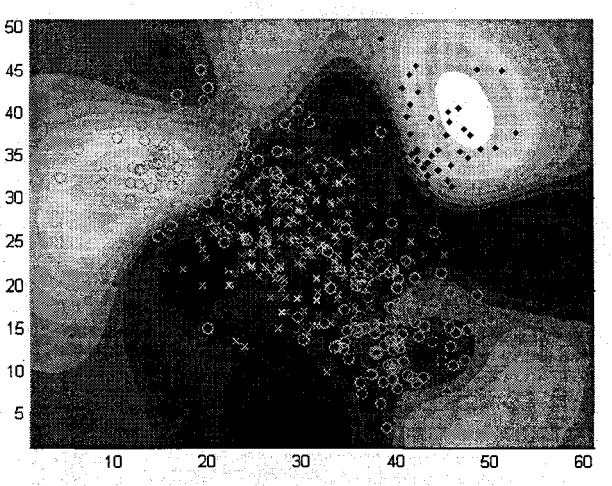
Les performances empiriques des approches locale et globale sont donc presque égales. Ceci confirme que l'optimisation individuelle des classifieurs produit un système multi-classe optimisé [1].

Bien que les exemples des classes marginales sont aussi pris en compte dans l'optimisation globale, les frontières de décision produites sont similaires (excepté pour $(o, \times)$) à celles de l'approche locale. En pratique, la procédure globale est beaucoup moins rapide que la procédure locale. Cependant, l'optimisation s'accompagne d'une réduction simultané des erreurs de validation de l'ensemble des classifeurs de l'ensemble. L'implantation de la procédure sur des machines parallèles peut d'ailleurs être un moyen efficace de gérer la complexité de calcul de la méthode.

[1] Bien qu'évidente, cette constatation a besoin d'une analyse théorique plus poussée. En particulier, l'erreur quadratique proposée dans l'approche globale pourrait être décomposée et comparée avec les termes d'erreur des classifieurs pris individuellement.



(a) avec optimisation locale.



(b) avec optimisation globale

Figure 35 Frontières obtenues pour le couple de classes $(o, \times)$

Le taux de reconnaissance final sur l'ensemble de test est de 84.58% avec l'optimisation globale. Celui obtenu avec l'optimisation locale est 85% (un seul faux positif de plus pour l'approche globale).

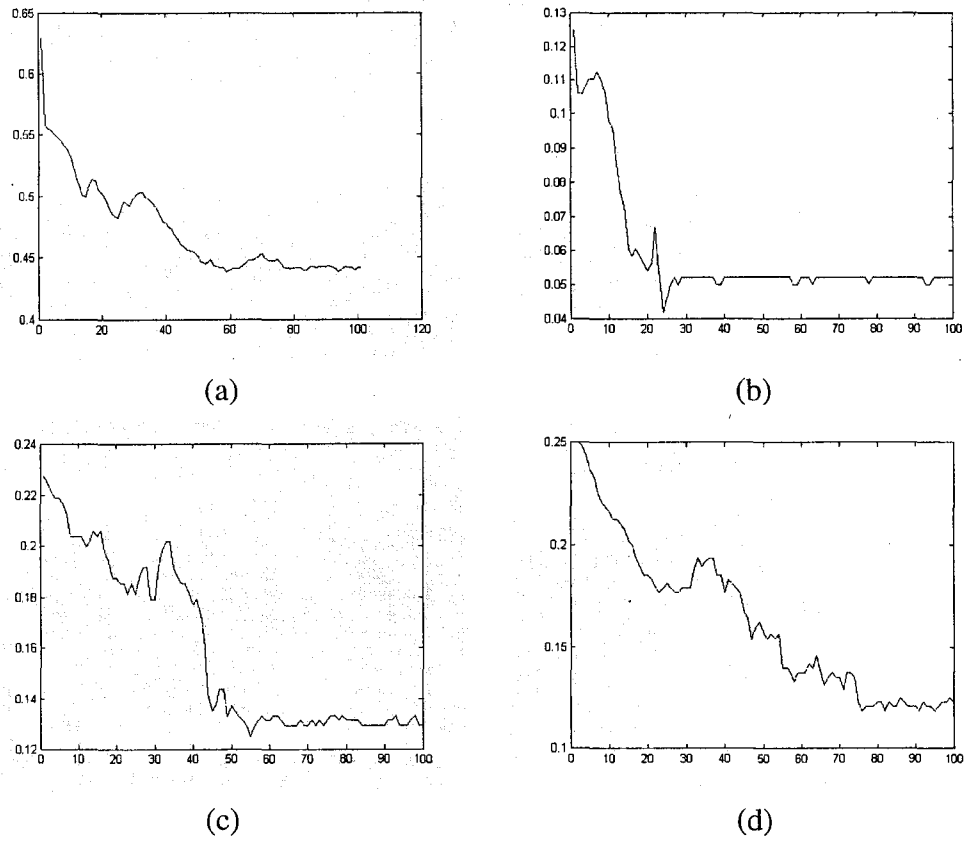


Figure 36 (a) Variation de la fonction objective de l'optimisation globale ; (b) variation de l'erreur de validation pour les classes $(\bullet, o)$; (c) variation de l'erreur de validation pour les classes $(\bullet, \times)$; (d) variation de l'erreur de validation pour les classes $(\times, o)$

8.2 Reconnaissance d'images de chiffres manuscrits arabes

Dans ce qui suit, nous discutons la partie expérimentale de la thèse sur une application réelle. Cette section est divisée en deux parties. Dans la première nous présentons les résultats d'expérience sur la base de données de chiffres arabes de USPS. Dans la deuxième, nous présentons les résultats d'expérience sur la base de données de chiffres Indiens, IND-CENPARMI.

8.2.1 La base de données de USPS

La base de données de USPS (de l'anglais «US postal Service») est un benchmark bien connu au sein de la communauté de reconnaissance de formes. Cet ensemble contient 9298 images de chiffres manuscrits dont 7291 images d'apprentissage et 2007 images de test. Ces images ont été saisies à partir d'images d'enveloppes collectées au centre CEDAR à Buffalo (États Unis) [54]. Chaque image de chiffre est représentée par 16×16 pixels de niveau de gris allant de 0 à 255 (Figure 37). Il est connu que l'ensemble de test de USPS est plutôt difficile - l'erreur humaine se situe autour de 2.5% [20]. Notons aussi qu'une partie des résultats réalisés sur cette base ont été obtenus avec un ensemble d'apprentissage amélioré. Par exemple, Drucker, Schapire et Simard avaient utilisées un ensemble d'apprentissage élargie de 9709 images contenant des images de chiffres imprimées qui améliore l'erreur de test [30]. Ceci a pour conséquence de changer le problème et biaise ainsi les résultats rendant leur interprétation difficile. De manière similaire, Bottou et Vapnik ont utilisé un ensemble augmenté de 9840 en rajoutant des images de chiffres imprimés. Dans la mesure ou l'ensemble de test ne contient pas d'images de ce genre, l'amélioration du taux de reconnaissance noté par les auteurs peut être due aux types d'images rajoutées qui sont assez différents des images originales de la base [19].

Par ailleurs, afin d'éviter de biaiser cette partie du travail, nous nous restreignons à la formulation classique du SVM et aux images originales de USPS. Aucun rajout d'images n'est effectué. Dans ce qui suit, nous démontrons l'efficacité de notre méthodologie de sélection de modèle en la comparant à une procédure de sélection manuelle des hyperparamètres (paramètres du noyau et C).

Les résultats de référence sur cette base sont rapportées dans le tableau XVII.

Pendant ce protocole, nous avons considéré la stratégie d'apprentissage en un-contre-un, où chaque classifieur $c(i, j)$ tente d'approximer la fonction de décision séparant les classes i et j. Les autres classes du corpus d'apprentissage sont alors omises et ne sont considé-

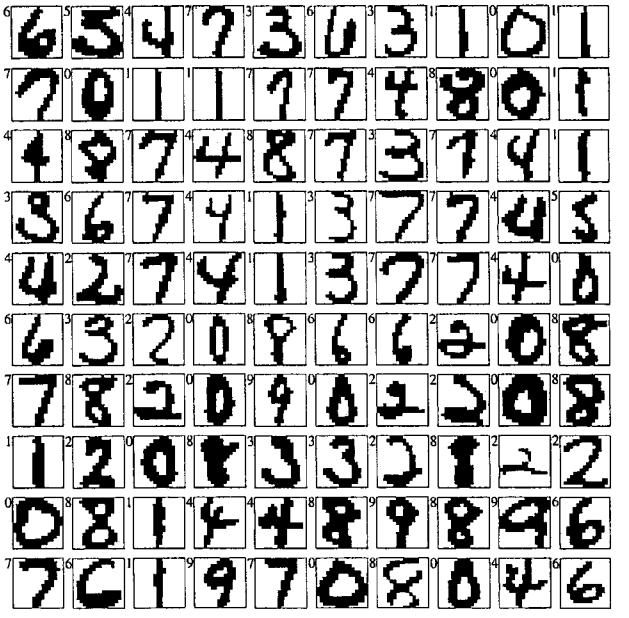


Figure 37 Échantillon représentant la première centaine d'images binarisées et leurs étiquettes dans la base de données USPS

Tableau XVII

Résultats de référence sur la base de données USPS

Classifieur	Erreur	ref.
Humain	2.5%	[20]
Tangent distance	2.6%	[95]
Perceptron à 2 couches	5.9%	[115, 87]
LeNet1	5.0%	[54]
SVM POLY	4.0%	[89]
SVM tanh	4.1%	[89]
SVM RBF	4.3%	[89, 36]
SVM Virtuel RBF	3.2%	[36]
SVM Virtual Poly	3.2%	[90]
noyau local	3.0%	[92]
k-means RBF	6.7%	[36]
Réseau RBF	4.9%	[36]
PMC	5.9%	[95]

rées que pour l'apprentissage des autres classifieurs. Dans le chapitre 3 (section 3.11), nous avons formulé l'approche d'apprentissage en un-contre-un et présenté les différents algorithmes de vote existants. Nous avons mis en revue différents schémas de normalisation des sorties des classifieurs en vue de prédire de l'appartenance d'une observation de test donnée.

Le tableau XVIII présente les erreurs de test en fonction des paramètres du noyau KMOD pour une valeur de C égale à 1000. Les résultats sont présentés pour quatre schémas de vote différents à savoir **PWC1**, **PWC4**, **PWC5** et **OPWC**. Ces derniers sont les fonctions d'activation qui donnent les meilleurs résultats. Notons que les sorties du SVM sont normalisées par rapport à l'amplitude de la marge comme décrit dans le chapitre 3. Dans ce tableau, nous avons échantillonné les valeurs de γ à trois valeurs, soient 0.1, 0.5 et 1.0. Pour des valeurs beaucoup plus grandes, le noyau tend vers un dirac et le SVM produit énormément de vecteurs de support. Pour σ, nous considérons les valeurs 0.5, 1.0, 5.0 et 10.0. Le meilleur résultat est obtenu pour $\sigma = 0.5$ où le taux d'erreur avoisine le 4.83%. Nous procédons à une autre sélection plus raffinée a travers une grille de valeurs différentes. Les résultats sont présentés dans le tableau XIX.

Dans le tableau XIX, nous procédons à un autre échantillonnage des paramètres du noyau autour de $\sigma = 0.5$. Nous considérons alors les valeurs 0.1, 0.2 et 0.3 pour γ et 2,3,4,5,6,7,8 et 9 pour σ. Les erreurs de test sont cette fois présentées pour $C = 10$ et $C = 1000$. Les résultats sont sensiblement similaires avec les deux valeurs de C. Les classes semblent donc séparables deux à deux. En effet, le cas échéant, l'influence de C sur la précision et le nombre de vecteur de support est négligeable. Notons que l'erreur de classification est définie :

$$\frac{\sum_{i=1}^{N}[l(x_i) \neq c_i]}{N},$$

où N représente le nombre d'observations de test x_i, $l(x_i)$ représentant le vote issu de l'ensemble de SVMs, c_i est la vraie classe de l'observation et $[z]$ étant l'opérateur logique

Tableau XVIII

Erreurs de classification en pourcentage sur USPS avec KMOD et C=1000

(γ,σ)	PWC1	PWC4	PWC5	OPWC
(0.1,0.5)	12.5	12.5	12.5	13.0
(0.1,1.0)	7.82	7.82	7.82	8.97
(0.1,5.0)	4.83	4.83	4.68	5.13
(0.1,10.0)	5.03	5.03	4.78	5.08
(0.5,0.5)	14.70	14.75	14.75	16.00
(0.5,1.0)	8.12	8.07	8.12	9.11
(0.5,5.0)	4.83	4.83	4.68	5.13
(0.5,10.0)	5.08	5.03	4.83	5.08
(1.0,0.5)	63.77	63.77	63.77	66.32
(1.0,1.0)	9.32	9.32	9.32	10.66
(1.0,5.0)	4.83	4.83	4.68	5.13
(1.0,10.0)	5.08	5.03	4.83	5.08

défini par :

$$[z] = \begin{cases} 1 & si\ z\ est\ vrai \\ 0 & sinon \end{cases}$$

Le meilleur taux d'erreur atteint est égal à 4.38% pour $C = 1000$. Pour $C = 10$, nous obtenons une valeur de 4.43%. Les deux valeurs sont obtenues pour le type de vote $PWC5$ et pour (γ,σ)= (0.2,6.0) et (γ,σ)=(0.3,6.0). PWC4 donne des résultats légèrement inférieurs à ceux de PWC5 mais supérieurs à ceux de PWC1. Ce dernier, utilisant la distance de Hamming, est le schéma le plus utilisé dans l'arbitrage un-contre-un.

Nous présentons dans le tableau XX le nombre moyen de vecteurs de support par classifieur. Ce nombre dépend inversement de la valeur de σ. La valeur de γ semble aussi influencer légèrement le nombre de vecteurs de support. Cependant, les données de USPS sont insensibles aux valeurs de C qui est pourtant un paramètre assez critique sur d'autres problèmes avec la stratégie un-contre-tous. En réalité la décomposition des données selon la stratégie un-contre-un réduit la complexité de la fonction de décision et ne produit pas

Tableau XIX

Erreurs de classification en pourcentage sur USPS avec KMOD en fonction de γ et σ pour C=1000 et C=10

	$C = 1000$				$C = 10$			
(γ, σ)	PWC1	PWC4	PWC5	OPWC	PWC1	PWC4	PWC5	OPWC
(0.1,2.0)	5.93	5.93	5.97	6.23	5.83	5.83	5.88	6.18
(0.1,3.0)	5.33	5.28	5.23	5.63	5.28	5.23	5.23	5.58
(0.1,4.0)	4.93	4.98	4.83	5.28	4.93	4.98	4.88	5.23
(0.1,5.0)	4.83	4.83	4.68	5.13	4.78	4.78	4.73	5.23
(0.1,6.0)	4.63	4.58	4.43	4.98	4.58	4.53	4.48	4.98
(0.1,7.0)	4.78	4.68	4.53	4.88	4.68	4.58	4.58	4.98
(0.1,8.0)	5.03	4.88	4.88	4.83	4.93	4.83	4.93	4.93
(0.1,9.0)	5.23	5.08	4.98	4.83	5.03	4.93	4.88	4.78
(0.2,2.0)	5.93	5.93	5.97	6.23	5.83	5.83	5.88	6.18
(0.2,3.0)	5.33	5.28	5.23	5.63	5.28	5.23	5.23	5.58
(0.2,4.0)	4.93	4.98	4.83	5.28	4.93	4.98	4.88	5.23
(0.2,5.0)	4.83	4.83	4.68	5.13	4.83	4.83	4.78	5.23
(0.2,6.0)	4.58	4.53	4.38	4.98	4.53	4.48	4.43	4.98
(0.2,7.0)	4.73	4.63	4.53	4.88	4.68	4.58	4.58	4.98
(0.2,8.0)	5.18	5.03	4.93	4.83	5.08	4.88	4.98	4.93
(0.2,9.0)	5.28	5.18	5.03	4.83	5.13	4.98	4.98	4.88
(0.3,2.0)	5.93	5.93	5.97	6.23	5.88	5.88	5.92	6.18
(0.3,3.0)	5.33	5.28	5.23	5.63	5.28	5.23	5.23	5.58
(0.3,4.0)	4.93	4.98	4.83	5.28	4.93	4.98	4.88	5.23
(0.3,5.0)	4.83	4.83	4.68	5.13	4.83	4.83	4.78	5.23
(0.3,6.0)	4.58	4.53	4.38	4.98	4.53	4.48	4.43	4.98
(0.3,7.0)	4.73	4.63	4.53	4.88	4.68	4.58	4.58	4.98
(0.3,8.0)	5.08	4.93	4.88	4.83	5.03	4.83	4.93	4.93
(0.3,9.0)	5.18	5.08	4.98	4.83	5.08	4.93	4.93	4.88

d'erreurs d'apprentissage. Auquel cas, la valeur de C n'influence pas la solution du SVM. Aux valeurs optimales du noyau, le SVM produit 178 vecteurs de support environ.

Tableau XX

Variation du nombre moyen de vecteurs de support sur USPS pour le noyau KMOD en fonction de γ et σ pour $C = 10$ et $C = 1000$

(γ,σ)	C=10	C=1000
(0.1,2.0)	562	562
(0.1,3.0)	371	371
(0.1,4.0)	273	273
(0.1,5.0)	215	215
(0.1,6.0)	178	178
(0.1,7.0)	154	154
(0.1,8.0)	138	138
(0.1,9.0)	125	125
(0.2,2.0)	564	563
(0.2,3.0)	371	370
(0.2,4.0)	273	272
(0.2,5.0)	215	214
(0.2,6.0)	178	177
(0.2,7.0)	154	154
(0.2,8.0)	138	138
(0.2,9.0)	125	125
(0.3,2.0)	565	564
(0.3,3.0)	372	370
(0.3,4.0)	273	272
(0.3,5.0)	215	214
(0.3,6.0)	178	177
(0.3,7.0)	154	154
(0.3,8.0)	138	137
(0.3,9.0)	125	124

Le tableau XXI présente les taux d'erreurs de test pour différentes valeurs de σ du RBF et différents schémas de vote du SVM. Le meilleur taux d'erreur trouvé est égal à 4.58% pour $C = 1000$. Pour $C = 10$, nous obtenons une valeur de 4.73%. Ces deux valeurs sont obtenues avec $PWC5$ et pour σ autour de 5. Dans le tableau XXII est présenté le nombre moyen de vecteurs de support par classifieur. Notons que le nombre de vecteurs de support à la valeur optimale de σ est égale à 216 environ. Ce nombre est nettement supérieur à celui produit avec le noyau KMOD.

Tableau XXI

Erreurs de classification en pourcentage sur USPS avec RBF en fonction de σ pour $C = 1000$ et $C = 10$

	$C = 1000$				$C = 10$			
σ	PWC1	PWC4	PWC5	OPWC	PWC1	PWC4	PWC5	OPWC
20.0	6.13	6.07	6.13	5.98	6.13	6.18	6.38	6.53
10.0	5.13	5.18	5.18	5.18	5.08	5.08	5.18	5.13
6.67	4.88	4.88	4.68	4.73	4.83	4.73	4.73	4.83
5.0	4.73	4.78	4.58	4.93	4.78	4.78	4.73	4.98
4.0	4.78	4.78	4.78	5.08	4.73	4.73	4.78	5.13
3.33	5.28	5.28	5.23	6.07	5.23	5.23	5.23	6.03
2.86	8.02	8.02	7.97	9.52	7.97	7.97	7.97	9.52
2.5	13.4	13.4	13.4	15.29	13.35	13.35	13.35	15.30
2.22	20.93	20.93	20.93	23.12	20.88	20.88	20.88	23.02
2.0	28.45	28.45	28.45	30.34	28.40	28.40	28.40	30.29

Tableau XXII

Variation du nombre moyen de vecteurs de support sur USPS pour le noyau RBF en fonction de σ pour $C = 1000$ et $C = 10$

γ	20.0	10.0	6.67	5.0	4.0	3.33	2.86	2.5	2.22	2.0
$C = 1000$	75	95	138	216	355	550	771	952	1086	1175
$C = 10$	97	97	139	217	357	552	772	954	1087	1175

Le tableau XXIII montre les taux d'erreurs pour différentes valeurs de d et différents schémas de vote des classifieurs. Le meilleur taux d'erreur obtenu est 5.33% pour les deux valeurs de C. Notons d'ailleurs que les performances coïncident exactement entre ces deux valeurs lorsque d est supérieur à deux. Les classes sont donc séparables pour ces valeurs de d. Le tableau XXIV présente la valeur moyenne du nombre de vecteurs de support par classifieur en fonction de d et C.

Tableau XXIII

Erreurs de classification en pourcentage sur USPS avec le noyau polynomial en fonction de d pour $C = 1000$ et $C = 10$

	$C = 1000$				$C = 10$			
d	PWC1	PWC4	PWC5	OPWC	PWC1	PWC4	PWC5	OPWC
1	7.27	7.07	11.06	7.32	7.22	7.07	10.91	7.37
2	5.48	5.38	9.21	5.33	5.48	5.38	9.22	5.33
3	5.93	5.73	11.51	6.43	5.93	5.73	11.51	6.43
5	7.97	7.97	14.99	27.20	7.97	7.97	14.99	27.20
7	11.36	11.21	17.28	45.89	11.36	11.21	17.28	45.89

Le nombre total de vecteurs de support pour le noyau linéaire avec $C = 1000$ est égal à 2919. Pour $C = 10$, ce nombre est égal à 2932. Cette différence indique qu'il existe au moins un modèle pour lequel les données ne sont pas linéairement séparables. En effet le modèle (3,5) est celui qui présente le plus de difficulté vu la similarité des classes '3' et '5'. Dans ce cas-ci, la performance du noyau polynomial est assez faible en comparaison avec le RBF et KMOD.

Le but des expériences dressées dans cette partie est de présenter les performances du SVM avec différents noyaux et schémas de vote sur les données de USPS. En particulier, on a trouvé que le noyau KMOD présente une erreur de 4.38% (pour C=1000), alors que les noyaux RBF et polynomial (d=2) donnent respectivement 4.58% et 5.33%. En terme de complexité du classifieur, le noyau KMOD aux valeurs optimales de ses paramètres, produit une moyenne de 177 vecteurs de support par classifieur. Ce nombre est beaucoup plus

Tableau XXIV

Variation du nombre moyen de vecteurs de support sur USPS pour un noyau Polynomial en fonction de d pour $C = 1000$ et $C = 10$

d	1	2	3	5	7
$C = 1000$	64	94	106	128	160
$C = 10$	64	94	106	128	160

faible que celui produit avec le RBF. Dans le tableau XVII, Scholkopf et al. obtiennent pour le RBF un taux d'erreur de 4.3% avec un SVM en stratégie un-contre-tous. Cette valeur est légèrement meilleure que celle que nous obtenons avec le noyau RBF en stratégie un-contre-un. Dépendamment des données, les deux stratégies de décomposition peuvent se comporter différemment. Aujourd'hui, il n'existe pas encore de méthode permettant de choisir a priori la meilleure stratégie de décomposition des classes.

Enfin, notons que la sélection manuelle (comme celle dans le tableau XVII) est réalisée sur la base de l'ensemble de test. Ceci favorise quelque peu la comparaison de la sélection manuelle par rapport à la sélection automatique.

8.2.2 Minimisation de l'erreur empirique

Nous discutons dans cette partie le protocole expérimental choisi pour valider le schéma d'optimisation des paramètres du noyau par minimisation de l'erreur empirique.

Nous avons départagé l'ensemble d'apprentissage original en deux partitions que nous avons appelées partition d'apprentissage et partition de validation. La taille de la partition d'apprentissage est 5291. La partition de validation est composée des 2000 derniers exemples de l'ensemble d'apprentissage. Les tailles des deux partitions ont été décidées de manière à allouer approximativement un tiers des données à l'ensemble de validation et deux tiers des données à l'ensemble d'apprentissage. Par ailleurs, comme nous l'avons déjà vérifié sur le problème XOR dans le chapitre 7, la taille de l'ensemble de validation

peut affecter les résultats et les performances du classifieur. Il est évident qu'une validation croisée ou une procédure de «Leave-One-Out» par exemple ne peut qu'améliorer la précision de la probabilité d'erreur estimée. Mais ceci est très prohibitif en temps de calcul. Nous nous assurons toutefois de choisir un nombre suffisant de données de validation.

Il est nécessaire d'initialiser les paramètres du noyau à une valeur raisonnable qui assure que la fonction objective de l'apprentissage du SVM soit convexe, et donc qu'une solution existe. Une trop faible valeur de σ (très grande valeur de γ) fait que l'ensemble des données sont des vecteurs de support, et la fonction objective reste sur un plateau dont il est difficile de se départir. Une valeur trop grande de σ (très faible valeur de γ) peut empêcher la convergence lors de l'apprentissage du SVM à cause de la capacité réduite du classifieur. Ceci est particulièrement vrai lorsque la valeur de C est assez grande.

Cependant, il est assez facile de choisir des valeurs de paramètres du noyau qui produisent un sur-apprentissage des données sans que l'on soit sur un plateau de l'erreur empirique. Ceci est spécialement le cas quand une grande proportion des données d'apprentissage sont des vecteurs de support (non pas la totalité).

Dans la validation expérimentale, nous avons utilisé la stratégie d'optimisation locale décrite dans le chapitre 6. Les modèles sont optimisés de façon séquentielle l'un après l'autre. Les hyper-paramètres de l'ensemble des classifieurs sont initialisés aux mêmes valeurs. Comme décrit précédemment, l'erreur empirique dans l'approche locale traduit la probabilité d'erreur estimée pour les exemples appartenant au couple de classes considéré.

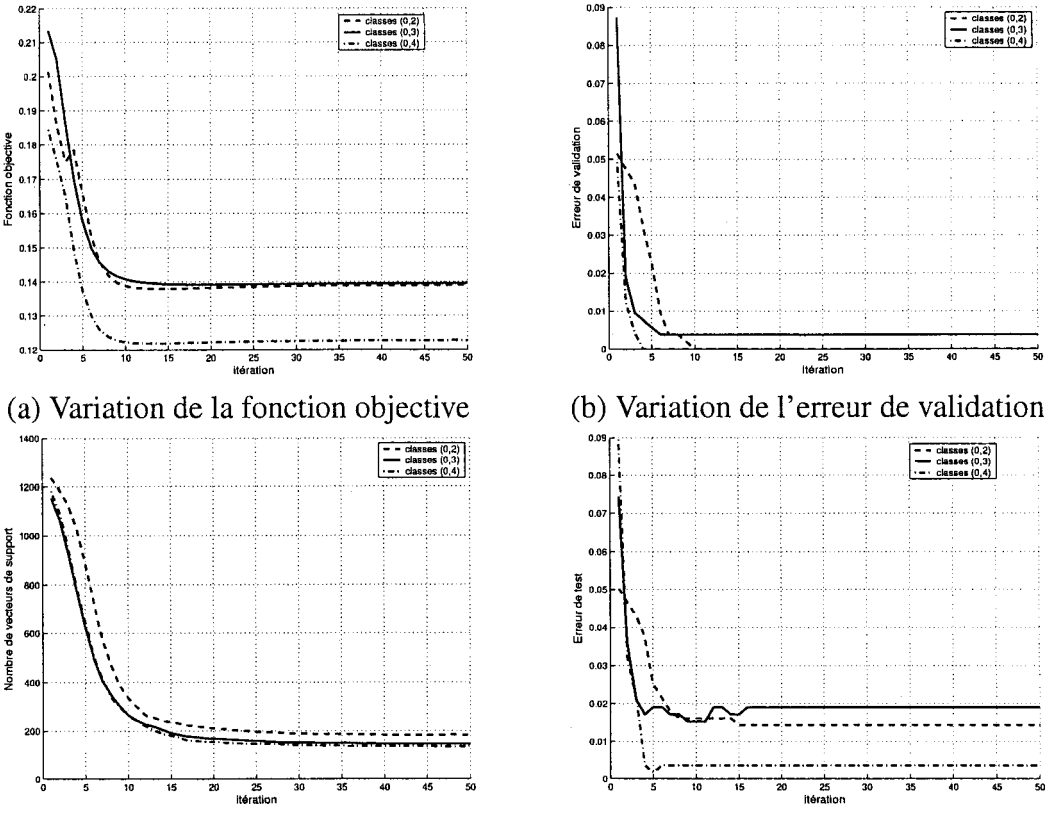
Nous avons implanté trois variantes de l'algorithme de minimisation basées respectivement sur une simple descente de gradient, le momentum et le quasi-Newton. Les courbes de la figure 38 montrent les variations de la fonction objective (courbe (a)), de l'erreur de validation (courbe (b)), du nombre de vecteurs de support (courbe (c)) et de l'erreur de test (courbe (d)) pour les couples de classes (0,2),(0,3) et (0,4) pour un noyau RBF ($\sigma_0 = 2.0$) et $C = 1000$.

La variation de la fonction objective montre qu'elle est bel et bien minimisée pour les trois couplets de classes considérés. Les courbes sont exemptes de bruit. Les fonctions objectives sont vraisemblablement convexes. Ceci s'accompagne d'une réduction considérable de l'erreur de validation (courbe (b)). Ainsi, à la 10ème itération les erreurs de validation sont réduites à 0% pour les classes (0,2) et (0,4), et à 0.48% pour le couplet (0,3).

La variation du nombre de vecteurs de support dans la courbe (c), montre deux constations :

1. la minimisation de la probabilité d'erreur est proportionnelle à celle du nombre de vecteurs de support.
2. le nombre de vecteurs de support continue à décroître alors même que l'erreur de validation est à son minimum.

En effet, la deuxième remarque est fort intéressante. Elle augure de façon claire que la minimisation de la probabilité de l'erreur réduit l'erreur de généralisation et la complexité du classifieur. Une procédure de sélection de modèle manuelle par exemple, pourrait se contenter du modèle du couplet (0,4) produit à l'itération 5, puisque celui ci produit zéro erreurs de validation. Or, le nombre de vecteurs de support à cette itération est $\simeq 600$, soit 4.5 fois plus important que le nombre de vecteurs de support à la convergence du critère. Les taux d'erreur de test de la courbe (d) montrent que ceux-ci sont aussi substantiellement réduits pour les couplets de classes (0,2), (0,3) et (0,4) pour finir aux valeurs 1.44%, 1.90% et 0.36% respectivement. Notons aussi que l'erreur de test continue de changer au delà de la 10ème itération alors que les erreurs de validation demeurent constantes.



(a) Variation de la fonction objective

(b) Variation de l'erreur de validation

(c) Variation du nombre de vecteurs de support

(d) Variation de l'erreur de test

Figure 38 Résultats d'optimisation sur USPS pour les modèles (0,2), (0,3) et (0,4) avec la méthode de descente de gradient (RBF, C=1000)

Les tableaux XXV et XXVI montrent respectivement les taux d'erreur de classification binaire des classifieurs avant et après minimisation de la fonction objective (probabilité d'erreur). Notons que l'erreur de test est réduite sur l'ensemble des classifieurs considérés. Cette réduction peut être importante comme celles enregistrées pour les classifieurs (2,3) et (2,5) ou les erreurs sont réduites de 25.55% à 2.47% et de 35.20% à 1.12% respectivement. Le nombre de vecteurs de support pour les mêmes couplets de classes varie de 947 à 198 et de 892 à 205 respectivement (voir tableaux XXVII et XXVIII). Les tableaux XXIX et XXX montrent respectivement les valeurs initiales et finales de la fonction objective pendant l'optimisation.

Tableau XXV

Erreur de test en pourcentage sur USPS avant optimisation avec C=1000 et le noyau RBF)

Classes	1	2	3	4	5	6	7	8	9
0	3.05	5.03	7.43	8.94	24.08	10.02	6.92	19.24	7.65
1	-	4.11	4.42	4.31	4.25	4.15	4.38	4.42	4.31
2	-	-	25.55	22.36	35.20	14.67	10.43	29.95	11.20
3	-	-	-	3.83	11.96	10.42	9.27	22.89	10.50
4	-	-	-	-	6.39	9.46	8.36	19.67	10.34
5	-	-	-	-	-	10.30	9.12	21.78	10.09
6	-	-	-	-	-	-	7.26	7.44	6.34
7	-	-	-	-	-	-	-	8.63	3.70
8	-	-	-	-	-	-	-	-	8.75

Tableau XXVI

Erreur de test en pourcentage sur USPS après optimisation avec C=1000 et le noyau RBF

Classes	1	2	3	4	5	6	7	8	9
0	0.48	1.44	1.90	0.36	0.96	0.38	0.59	1.14	0.37
1	-	1.08	0.70	1.51	0.71	1.38	0.97	0.93	0.68
2	-	-	2.47	2.01	1.12	0.82	1.16	3.57	0.27
3	-	-	-	0.55	4.60	0.60	1.60	3.31	0.87
4	-	-	-	-	1.11	1.89	2.31	0.82	2.39
5	-	-	-	-	-	0.91	0.65	2.76	0.89
6	-	-	-	-	-	-	0.32	0.60	0.29
7	-	-	-	-	-	-	-	1.60	3.40
8	-	-	-	-	-	-	-	-	1.46

L'histogramme (a) de la figure 39 compare les erreurs de test avant et après optimisation pour les modèles $(0,1),(0,2),\ldots,(0.9)$. L'histogramme (b) de la figure 39 compare le nombre de vecteurs de support des mêmes modèles avant et après leur optimisation. Il est à noter que les nombres de vecteurs de support montrés sont produits sur la partition d'apprentissage et non pas l'ensemble d'apprentissage original. Le nombre de vecteurs de

Tableau XXVII

Nombre de vecteurs de support sur USPS avant optimisation avec C=1000 et le noyau RBF

Classes	1	2	3	4	5	6	7	8	9
0	899	1236	1153	1181	1198	1284	1203	1215	1235
1	-	598	523	555	468	555	473	486	504
2	-	-	947	977	892	982	900	909	931
3	-	-	-	902	817	907	825	834	855
4	-	-	-	-	847	937	857	864	893
5	-	-	-	-	-	852	769	779	800
6	-	-	-	-	-	-	859	869	890
7	-	-	-	-	-	-	-	787	826
8	-	-	-	-	-	-	-	-	818

Tableau XXVIII

Nombre de vecteurs de support sur USPS après optimisation avec C=1000 et le noyau RBF

Classes	1	2	3	4	5	6	7	8	9
0	58	184	146	134	205	187	96	151	108
1	-	92	74	95	73	90	61	96	80
2	-	-	198	242	205	226	148	219	152
3	-	-	-	127	232	123	119	197	141
4	-	-	-	-	174	177	199	182	227
5	-	-	-	-	-	195	130	204	148
6	-	-	-	-	-	-	99	144	102
7	-	-	-	-	-	-	-	135	235
8	-	-	-	-	-	-	-	-	175

support obtenus après entraînement du système optimisé sur l'ensemble d'apprentissage original est plus important. Cependant, l'erreur de test des classifieurs est plus faible.

Tableau XXIX

Fonction objective ($\times 10^{-4}$) sur USPS avant optimisation avec C=1000 et le noyau RBF

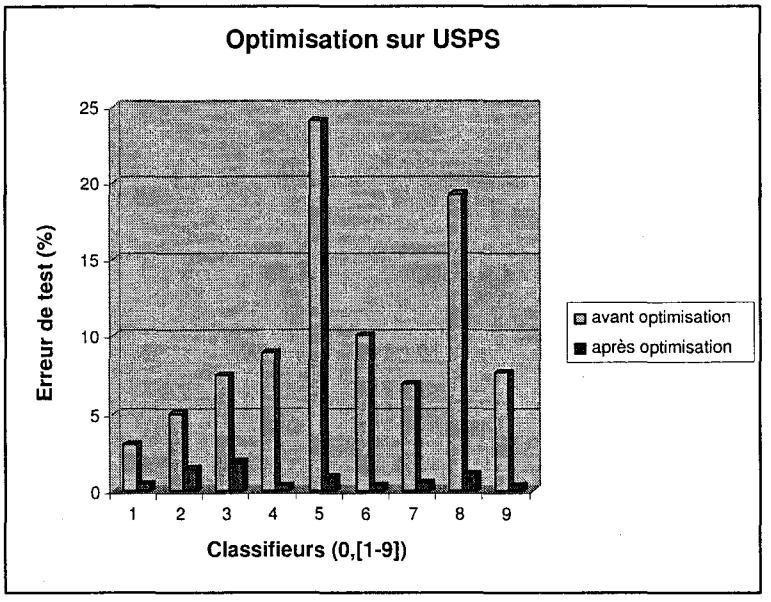
Classes	1	2	3	4	5	6	7	8	9
0	1386	2015	2135	1842	2223	2043	1901	1875	1756
1	-	1385	1383	1361	1321	1284	1276	1242	1194
2	-	-	3205	2789	3165	2561	2354	2818	2283
3	-	-	-	2842	3161	2543	2268	2836	2216
4	-	-	-	-	3013	2554	2368	2730	2420
5	-	-	-	-	-	2784	2469	3059	2378
6	-	-	-	-	-	-	2253	2533	2274
7	-	-	-	-	-	-	-	2456	2299
8	-	-	-	-	-	-	-	-	2428

Tableau XXX

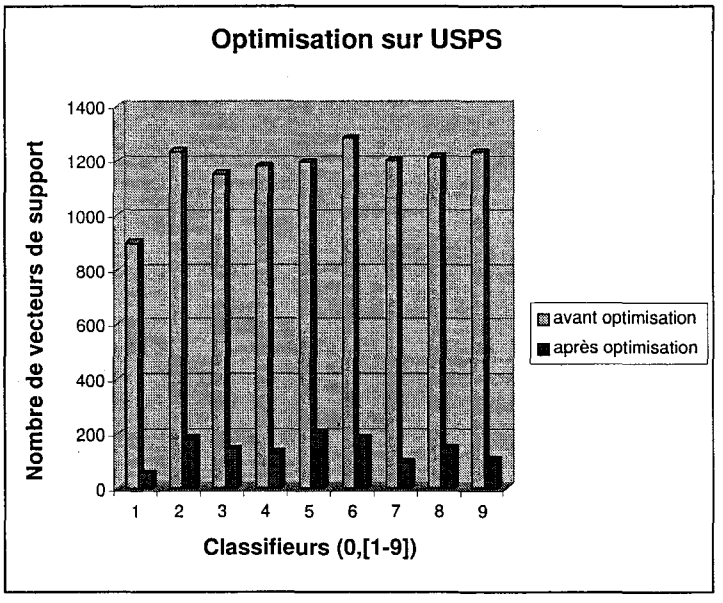
Fonction objective ($\times 10^{-4}$) sur USPS après optimisation avec C=1000 et le noyau RBF

Classes	1	2	3	4	5	6	7	8	9
0	1269	1390	1395	1228	1407	1253	1126	1183	1040
1	-	1255	1255	1233	1168	1149	1147	1126	1060
2	-	-	1556	1456	1539	1346	1366	1451	1261
3	-	-	-	1384	1725	1326	1400	1536	1319
4	-	-	-	-	1464	1379	1492	1464	1524
5	-	-	-	-	-	1558	1404	1681	1364
6	-	-	-	-	-	-	1321	1428	1326
7	-	-	-	-	-	-	-	1510	1642
8	-	-	-	-	-	-	-	-	1533

Les courbes de la figure 40 montrent les variations de la fonction objective (courbe (a)), de l'erreur de validation (courbe (b)), du nombre de vecteurs de support (courbe (c)) et de l'erreur de test (courbe (d)) pour les couples de classes (0,2),(0,3) et (0,4) avec la l'algorithme de Quasi-Newton pour le noyau RBF et $C = 1000$.



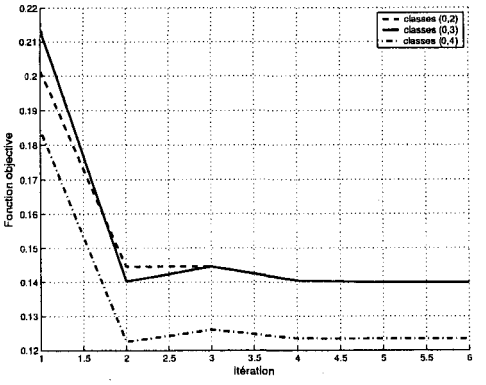
(a) Erreurs de test



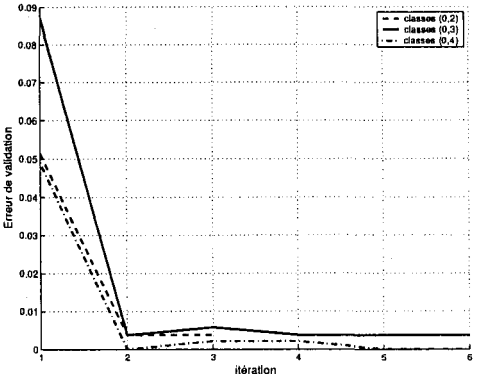
b) Nombre de vecteurs de support

Figure 39 Performances de certains classifieurs avant et après l'optimisation

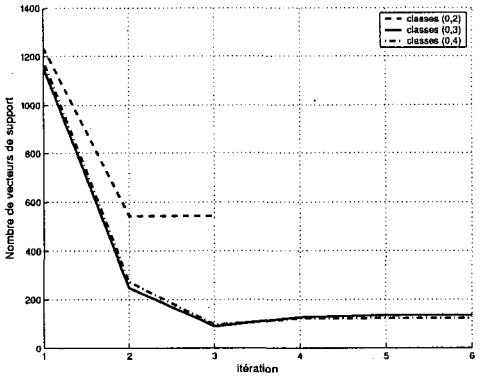
Les tableaux XXXI, XXXII et XXXIII montrent les taux de reconnaissance du système optimisé sur l'ensemble de test avec les noyaux KMOD, RBF et polynomial (d=2) respectivement. Les résultats sont obtenus pour quatre schémas de vote différents de l'ap-



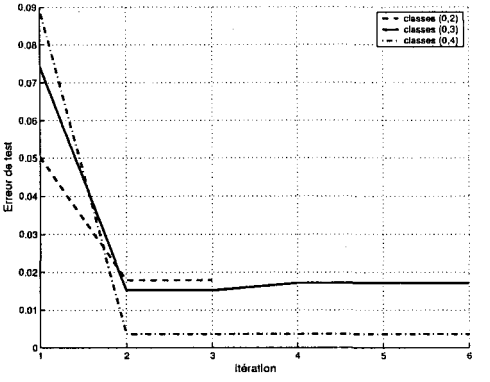
(a) Variation de la fonction objective



(b) Variation de l'erreur de validation



(c) Variation du nombre de vecteurs de support



(d) Variation de l'erreur de test

Figure 40 Résultats d'optimisation sur USPS pour les modèles (0,2), (0,3) et (0,4) avec la méthode de Quasi-Newton (RBF, C=1000)

proche un-contre-un présentés dans la section 3.11 du chapitre 3. Ce sont $PWC1$, $PWC4$, $PWC5$ ET $OPWC$. Nous avons omis de présenter les résultats avec $PWC2$ et $PWC3$ parce que leurs performances sont médiocres.

Les meilleurs taux d'erreur obtenus sont 4.2%, 4.4% et 5.3% pour les noyaux KMOD, RBF et polynomial (d=2) respectivement. Ces valeurs sont relativement meilleures pour la descente de gradient que pour la méthode de Quasi-Newton. Ces valeurs sont obtenues avec le schéma $PWC4$ (KMOD et RBF) et $OPWC$ (polynomial, d=2).

Comme on peut le remarquer, les taux d'erreur sont meilleurs que ceux obtenus pendant la sélection manuelle des hyper-paramètres (malgré le fait qu'elle soit réalisée sur l'en-

semble de test). Les valeurs de cette dernière sont des estimations optimistes à cause de l'utilisation des données de test pour le choix des hyper-paramètres.

L'avantage de la sélection automatique est de pouvoir adapter le profil du noyau aux données des classes considérées deux à deux. Les valeurs des hyper-paramètres retrouvées ne sont pas contraintes à être égales comme est dans le cas de la sélection manuelle. Ceci constitue un avantage indéniable de l'approche.

Tableau XXXI

Taux d'erreur sur l'ensemble de test du système optimisé sur USPS avec C=1000 et le noyau KMOD

	PWC1	PWC4	PWC5	OPWC
Descente de gradient	4.4	4.3	4.2	4.6
Quasi-Newton	4.9	4.7	4.6	4.8

Tableau XXXII

Taux d'erreur sur l'ensemble de test du système optimisé sur USPS avec C=1000 et le noyau RBF

	PWC1	PWC4	PWC5	OPWC
Descente de gradient	4.7	4.8	4.4	4.8
Quasi-Newton	5.2	5.2	5.1	5.0

Tableau XXXIII

Taux d'erreur sur l'ensemble de test du système optimisé sur USPS (C=1000, polynomial, d=2)

	PWC1	PWC4	PWC5	OPWC
Descente de gradient	5.4	5.4	5.3	5.5
Quasi-Newton	5.8	5.7	5.7	5.6

8.3 La base de données de chiffres indiens

La base de données de chiffres Indiens de CENPARMI a été collectée à partir d'images de chèques Saoudiens fournis par la compagnie «Al Rajhi Banking corporation». Ces images ont subi les traitements préliminaires nécessaires à l'extraction du tracé et l'enlèvement du fond de l'image. En fait, la base de chiffres indiens a été constituée à partir de la base de montants numériques dont on a extrait les images de chiffres. Aussi, chaque image de montant littéral est associée à une image de montant en chiffres de telle manière qu'une vérification post reconnaissance soit possible en appariant les résultats de reconnaissance de chacune des images s'il y a lieu. Les chiffres indiens sont utilisés dans la majeure partie du moyen orient et du sud-Ouest Asiatique. Dû au manque de données, peu de systèmes, sinon aucun ne traitent ce type d'images. Les chiffres indiens, hormis leur formes, sont codés en décimal de même que les chiffres arabes. On y dénombre dix classes de chiffres indiens (Figure 41).

Les images sont considérées après pré-traitement. Ainsi, le tracé est filtré afin d'enlever les segments ou les agrégats de pixels inutiles. Un épaississement du contour est par la suite réalisé afin de conserver des structures morphologiques importantes pour caractériser l'image.

Dans la littérature, plusieurs travaux ont porté sur l'élaboration de nouvelles primitives à même de procurer un pouvoir discriminant inter-classe accru tout en minimisant la variabilité intra-classe. Ces primitives sont généralement classées en deux familles : les primitives morphologiques (boucle, concavité, dépassements, points extrêmes du tracé, intersections, etc) et les primitives statistiques qui dérivent des mesures de distribution spatiale de pixels (zonage, moments invariants, descripteurs de Fourrier, etc). Les caractéristiques morphologiques et statistiques sont complémentaires dans la mesure ou deux sortes de propriétés sont mises en relief. Afin de caractériser nos images de chiffres, nous avons opté pour un vecteur de primitives hybride combinant des statistiques sur les pixels du contour et

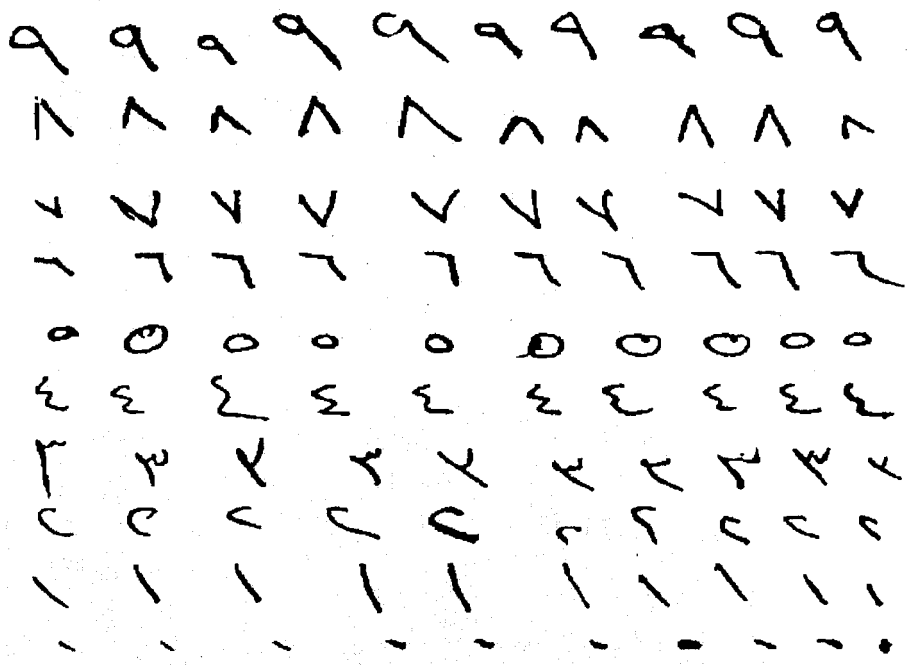
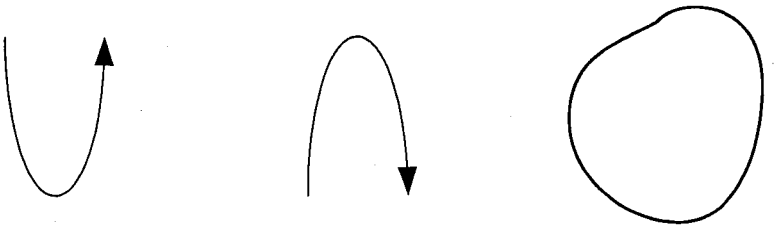


Figure 41 Échantillon de chiffres indiens

sur les pixels constituant certaines formes morphologiques. Ceci revient à opérer une reconnaissance grossière via les propriétés structurelles et une reconnaissance plus raffinée par analyse de la distribution spatiale des formes composant l'objet à reconnaître. Nous décrivons ci-dessous le jeu de primitives considérées.

8.3.1 Primitives perceptuelles

L'information locale comme celle contenue dans les gradients et les courbures de pixels du contour n'est pas suffisante pour représenter des motifs géométriques complexes et sujets au bruit. Aussi, l'aspect morphologique constitue un facteur très important dans l'interprétation humaine des images de chiffres. Nous avons donc décidé de considérer un premier jeu de primitives basé sur des caractéristiques structurelles calculées par rapport à quatre motifs morphologiques recherchés dans l'image. Cheriet et al. dans [25] et Strathy et al. dans [103] ont utilisé le concept de régions pour la segmentation de mots et des chaînes de



(a) région en vallée (b) région en montagne (c) région en boucle

Figure 42 Primitives structurelles considérées

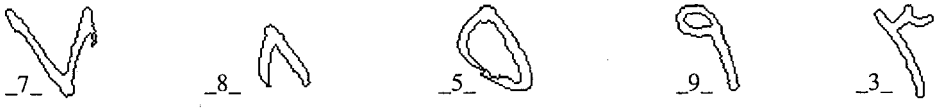


Figure 43 Chiffres indiens contenant des primitives discriminantes

chiffres. La caractérisation des images en analysant certaines formes géométriques a été aussi l'objet d'autres travaux tels que ceux présentés dans [26, 71, 60]. Heutte et al. dans [42] se sont assez étendus sur le sujet.

Nous avons observé que cinq classes parmi dix montrent des formes géométriques simples à calculer mais assez discriminantes. Ces chiffres correspondent aux classes '7', '8', '5', '9' et '3' qui contiennent respectivement une région en vallée, une région en montagne, une boucle, une boucle, et deux régions en vallée (Figures 42 et 43).

En combinant ces primitives morphologiques avec une grille de zonage, nous sommes à même de calculer la distribution des pixels des primitives morphologiques se retrouvant dans chaque zone de la grille. L'image est divisée en une grille de 4×4. Le nombre de pixels du contour dans chacune des zones, appartenant à chacune des quatre régions est déterminé. Il en résulte un vecteur de quatre valeurs associées à chaque zone de la grille.

8.3.2 Primitives statistiques

Les primitives statistiques considérées sont basées sur le calcul de la distribution des direction de Freeman et de courbure des pixels composant les contours du tracé. Le code

en chaîne du contour des caractères est calculé par un suivi en «run-length» [103] des images. Un adoucissement du contour est aussi réalisé en appliquant un moyenneur de longueur cinq à chaque pixel du contour. En associant la procédure de zonage citée plus haut, ce sous-ensemble de primitives devient un vecteur de treize valeurs représentant les statistiques sur les directions de Freeman et les courbures des pixels inclus dans chacune des zones définies par la grille englobante. Soit un vecteur de 13 * 16 = 208 valeurs. Les valeurs de courbure calculées sont quantifiées en cinq valeurs. L'ensemble des primitives est composé donc de (8+5+4) *16 = 272 valeurs. Afin de normaliser ces primitives, nous avons converti ces nombres en pourcentage du nombre de pixels total du contour dans chaque zone.

8.3.3 Expériences

La base de données INDCENPARMI contient deux corpus de données. La taille du corpus d'apprentissage est 4682. Celle de l'ensemble de test est 1939. Nous avons départagé l'ensemble des données d'apprentissage en deux partitions :

- une partition d'apprentissage de 3000 exemples,
- une partition de validation de 1682 exemples.

Nous avons alors suivi un protocole semblable à celui utilisé pour USPS. Nous avons procédé manuellement à une série d'expériences afin d'estimer les meilleurs hyper-paramètres de noyaux. Un intervalle pré-défini de valeurs de paramètres permettent une sélection de modèles manuelle pour l'ensemble de classifieurs. Encore faut-il choisir le bon intervalle de recherche. Les résultats obtenus représentent les erreurs de test de la combinaison de l'ensemble de classifieurs appris en stratégie un-contre-un sur l'ensemble de test. Les erreurs sur l'ensemble d'apprentissage étant souvent nulles, elles sont simplement omises. Les tableaux XXXIV et XXXV montrent les erreurs de test pour un noyau polynomial de degré 2 et 3 respectivement avec C=100. Les variables a et b du tableau représentent respectivement la constante multiplicative et la constante additive du noyau.

Tableau XXXIV

Taux d'erreurs sur INDCENPARMI avec le noyau polynomial pour d=2 et C=100

(a,b)	PWC1	PWC4	PWC5	OPWC
(0.1,0.1)	2.24	2.18	3.04	2.18
(0.1,0.5)	2.18	2.13	2.98	2.24
(0.1,1)	2.29	2.24	3.14	2.34
(0.1,5)	2.45	2.29	3.89	2.45
(0.1,10)	2.61	2.50	4.64	2.72
(0.5,0.1)	2.24	2.18	5.17	2.24
(0.5,0.5)	2.24	2.18	5.22	2.24
(0.5,1)	2.24	2.18	5.22	2.24
(0.5,5)	2.29	2.24	5.12	2.34
(0.5,10)	2.34	2.29	5.38	2.34
(1,0.1)	2.24	2.18	5.44	2.24
(1,0.5)	2.24	2.18	5.44	2.24
(1,1)	2.24	2.18	5.44	2.24
(1,5)	2.18	2.13	5.49	2.24
(1,10)	2.29	2.24	5.81	2.34
(5,0.1)	2.24	2.18	6.02	2.24
(5,0.5)	2.24	2.18	6.02	2.24
(5,1)	2.24	2.18	6.02	2.24
(5,5)	2.24	2.18	6.08	2.18
(5,10)	2.24	2.18	6.13	2.24
(10,0.1)	2.24	2.18	6.19	2.24
(10,0.5)	2.24	2.18	6.13	2.24
(10,1)	2.24	2.18	6.13	2.24
(10,5)	2.24	2.18	6.13	2.24
(10,10)	2.24	2.18	6.08	2.18

Le tableau de la figure XXXVI, représente les erreurs de test ainsi que le nombre de vecteurs de support retrouvés avec le noyau KMOD pour différentes valeurs de paramètres et $C = 100$.

Les tableaux XXXVIII et XXXIX montrent les taux d'erreur du système optimisé sur l'ensemble de test pour les noyaux KMOD et RBF. Les performances sont montrées en fonction de la méthode de minimisation utilisée (descente de gradient vs. quasi-Newton)

Tableau XXXV

Taux d'erreurs sur INDCENPARMI avec le noyau polynomial pour d=3 et C=100

(a,b)	PWC1	PWC4	PWC5	OPWC
(0.1,0.1)	2.02	1.92	4.16	2.45
(0.1,0.5)	2.08	1.97	4.42	2.29
(0.1,1)	2.18	2.08	4.58	2.29
(0.1,5)	2.34	2.29	5.22	2.24
(0.1,10)	2.45	2.29	5.65	2.40
(0.5,0.1)	2.02	1.92	5.54	2.45
(0.5,0.5)	2.02	1.92	5.54	2.45
(0.5,1)	2.02	1.92	5.65	2.40
(0.5,5)	2.18	2.08	5.92	2.29
(0.5,10)	2.24	2.18	6.08	2.29
(1,0.1)	2.02	1.92	5.65	2.45
(1,0.5)	2.02	1.92	5.70	2.45
(1,1)	2.02	1.92	5.76	2.45
(1,5)	2.08	1.97	6.13	2.29
(1,10)	2.18	2.08	6.08	2.29
(5,0.1)	2.02	1.92	5.76	2.50
(5,0.5)	2.02	1.92	5.76	2.45
(5,1)	2.02	1.92	5.76	2.45
(5,5)	2.02	1.92	5.92	2.45
(5,10)	2.02	1.92	6.02	2.40
(10,0.1)	2.02	1.92	5.76	2.50
(10,0.5)	2.02	1.92	5.76	2.50
(10,1)	2.02	1.92	5.76	2.45
(10,5)	2.02	1.92	5.81	2.45
(10,10)	2.02	1.92	5.92	2.45

et le schéma de combinaison (vote) des classifieurs (PWC1,PWC4,PWC5 et OPWC). Les valeurs obtenues sont pour C=100.

Nous avons aussi appliqué l'approche globale de minimisation sur les données INDCEN-PARMI tel que expliqué dans le chapitre 6 et la section 8.1. Nous présentons dans le ta-

Tableau XXXVI

Taux d'erreurs et nombre de vecteurs de support sur INDCENPARMI avec le noyau KMOD pour C=100

(γ, σ)	SV	PWC1	PWC4	PWC5	OPWC
(0.1,1)	414	3.46	3.46	3.46	4.64
(0.1,4)	89	1.92	1.92	1.92	2.08
(0.1,5)	72	1.97	1.97	1.97	2.08
(0.1,6)	63	1.97	1.97	1.97	2.13
(0.1,10)	46	2.18	2.24	2.24	2.18
(0.5,1)	438	3.68	3.68	3.68	4.90
(0.5,4)	89	1.92	1.92	1.92	2.08
(0.5,5)	72	1.97	1.97	1.97	2.08
(0.5,6)	63	1.97	1.97	1.97	2.13
(0.5,10)	46	2.13	2.02	2.02	2.24
(1,1)	520	4.64	4.64	4.64	6.40
(1,4)	90	1.92	1.92	1.92	2.08
(1,5)	73	1.97	1.97	1.97	2.08
(1,6)	63	1.97	1.97	1.97	2.13
(1,10)	46	2.08	1.97	1.97	2.24

Tableau XXXVII

Taux d'erreurs et nombre de vecteurs de support sur INDCENPARMI avec le noyau RBF pour C=100

γ	SV	PWC1	PWC4	PWC5	OPWC
0.05	38	2.35	2.24	2.24	2.40
0.10	41	2.29	2.19	2.19	2.35
0.15	49	2.03	2.03	2.03	2.19
0.20	60	1.97	1.97	2.03	2.08
0.25	77	1.92	1.92	1.92	2.08
0.30	103	1.97	1.97	1.97	2.13
0.35	144	2.13	2.13	2.13	2.24
0.40	210	2.08	2.08	2.08	2.67
0.45	298	2.61	2.61	2.61	3.68
0.50	401	4.22	4.22	4.22	5.92

Tableau XXXVIII

Taux d'erreurs du système optimisé sur INDCENPARMI avec C=100 et le noyau RBF

	PWC1	PWC4	PWC5	OPWC
Descente de gradient.	1.9	1.9	1.9	2.1
Quasi-Newton	2.2	2.2	2.2	2.8

Tableau XXXIX

Taux d'erreurs du système optimisé sur INDCENPARMI avec C=100 et le noyau KMOD

	PWC1	PWC4	PWC5	OPWC
Descente de gradient.	2.0	2.0	2.0	2.1
Quasi-Newton	2.3	2.3	2.2	2.9

bleau XL les taux d'erreur avec la méthode de momentum pour les noyaux RBF et KMOD, et pour C=100.

Notons que cette approche est très prohibitive en temps de calcul à cause que l'ensemble des observations de la partition de validation sont utilisées pour l'optimisation des hyper-paramètres des classifieurs. Du point de vue pratique, la complexité de cet algorithme varie de cinq à dix fois par rapport à celle de l'approche locale selon les valeurs initiales des hyper-paramètres.

Les performances sont cependant sensiblement similaires à celles de l'approche locale avec 2.0% d'erreur approximativement.

8.3.4 Analyse des résultats

Nous avons présenté les résultats d'expérience de l'optimisation des paramètres de noyaux de SVMs sur des données multiclasses. Nous avons organisé nos résultats en trois parties.

Tableau XL

Taux d'erreurs sur l'ensemble de test du système optimisé avec l'approche globale sur la base INDCENPARMI pour les noyaux KMOD et RBF et C=100

	PWC1	PWC4	PWC5	OPWC
KMOD	2.1	2.1	2.0	2.2
RBF	2.1	2.1	2.0	2.1

Dans la première, nous considérons un problème synthétique à trois classes et deux attributs. Ce problème à deux dimensions permet de visualiser les frontières de décision obtenues après optimisation des classifieurs. Dans cette expérience (et aussi dans le restant des expériences) les paramètres de noyaux sont initialisés de telle façon qu'un surapprentissage des données ait lieu. Les frontières de décision sont alors très bruitées et séparent mal les données de test. Pour un noyau RBF et avec C=100, l'approche locale d'optimisation sur ce corpus donne 85% de taux de reconnaissance sur l'ensemble de test alors que l'approche globale donne 84.58%. Les frontières de décision obtenues sont similaires excepté pour le couplet de classes $(o, \times)$ qui semblent légèrement différentes. Les performances des deux approches sont alors équivalentes. Aussi, ceci prouve que l'approche locale d'optimisation minimise l'erreur multiclasse. En terme de complexité, l'approche globale est prohibitive mais intéressante. Elle est intéressante parce qu'il est possible de paralléliser l'algorithme sur une machine parallèle rendant son calcul plus rapide. Aussi, les résultats empiriques que nous présentons ouvrent la porte à des considérations théoriques qu'il faut certainement explorer et prendre en compte.

Dans la deuxième partie du chapitre nous présentons les résultats d'expérience sur la base de données USPS. Le protocole expérimental est organisé en deux étapes. Dans la première, nous procédons à une sélection manuelle des paramètres de différents noyaux (KMOD, RBF, polynomial d=2 et d=3). La meilleure performance est obtenue avec le noyau KMOD avec 95.62% (4.38% d'erreur) contre 95.42% (4.58% d'erreur) pour le RBF pour C=100. Avec C=10, les résultats sont légèrement différents. Cependant, les

complexités des classifieurs sont sensiblement proches. Les performances avec le noyau polynomial sont par ailleurs mauvaises.

Sur USPS, l'optimisation automatique des hyper-paramètres est réalisée avec l'approche locale. Les valeurs des paramètres de noyaux sont trouvées de façon séquentielle en optimisant les 45 SVM du système de l'approche un-contre-un. Les résultats consolident les remarques observées sur le problème binaire du chapitre 7. En effet, la minimisation de l'erreur empirique estimée sur l'ensemble de validation, minimise non seulement l'erreur de classification, mais aussi la complexité du classifieur à travers le nombre de vecteurs de support. Dans les cas ou l'erreur de validation demeure inchangée pendant le processus de l'optimisation, on observe une réduction assez importante de la complexité comme montré dans la courbe (c) de la figure 38. Les taux de reconnaissance obtenus après l'entraînement du système avec les valeurs des hyper-paramètres finales des classifieurs sont 95.8%, 95.6% et 94.7% respectivement avec les noyaux KMOD, RBF et polynomial (d=2). Il est intéressant de noter l'amélioration des taux de reconnaissance (0.2%) par rapport à la sélection manuelle. Cette amélioration est due au nombre de degré de libertés de l'optimisation automatique qui cherche $45 \times n$ paramètres. La sélection manuelle ne cherche que n valeurs de paramètres (n étant le nombre de paramètres du noyau courant) en leur attribuant les mêmes valeurs à travers l'ensemble des 45 noyaux considérés.

L'autre grand apport de la procédure est la minimisation automatique de la complexité en réduisant considérablement le nombre de vecteurs de support. Ceci est remarqué quelque soit le type des données et la difficulté des classes à séparer. Comme montré dans l'histogramme (b) de la figure 39, lorsqu'un sur-apprentissage initial des données est présent, la méthode enlève jusqu'à 90% de vecteurs de support.

L'autre constatation intéressante concerne la performance du noyau KMOD. En effet, sur USPS, KMOD produit des classifieurs de complexité moindre comparativement au RBF [2].

[2] une différence de 30 vecteurs de support en moyenne environ, soit 20% approximativement. Voir tableaux XX et XXII

En effet, la non-linéarité supplémentaire du noyau KMOD permet d'interpoler facilement des frontières de décision bruitées comme celles induites par les valeurs de pixels. Par ailleurs, sur INDCENPARMI, KMOD produit un nombre plus important de vecteurs de support que le RBF. Celui-ci est un problème plus facile et le SVM opère sur des caractéristiques extraites moins clairsemées et plus localisées. La non-linéarité supplémentaire de KMOD est dans ce cas inutile et produit des classifieurs avec plus de vecteurs de support (Voir tableaux XXXVI et XXXVII).

Nous avons aussi pu établir l'efficacité de la sélection automatique des paramètres de noyaux sur les données de INDCENPARMI où nous avons pu obtenir les mêmes performances que celles obtenues avec la sélection manuelle. Sur ces données, la sélection manuelles des paramètres semblent montrer que les taux de reconnaissance sont sensiblement similaires pour les différents noyaux expérimentés. Ils sont de 98.1% pour les trois noyaux KMOD, RBF et polynomial (d=3). Pour le SVM polynomial (d=2) est de 97.9% approximativement. Cette relative insensibilité du SVM par rapport aux noyaux est due à la robustesse des caractéristiques extraites vis à vis du bruit présent dans les images. Sur USPS cependant, aucune procédure d'extraction de caractéristiques n'est réalisée, et l'espace d'entrée du SVM est constitué des niveaux de gris des pixels.

Enfin, les résultats du tableau XL, montrent que les performances du système obtenues avec l'approche locale sont voisines à celles obtenues avec l'approche globale. Seulement, cette dernière est prohibitive en espace mémoire et en complexité de calcul où les temps de simulation sur INDCENPARMI peuvent durer jusqu'à une vingtaine d'heures sur un PENTIUM III de 850MHZ.

8.4 Conclusion

Dans ce chapitre, nous avons dressé un protocole expérimental exhaustif pour valider la méthodologie d'optimisation des paramètres du noyau pour des données multiclasses dans la stratégie un-contre-un.

Dans un premier temps, les approches d'optimisation globale et locale sont validées et comparées sur un problème synthétique multiclasse à deux attributs.

Ensuite, nous considérons des données réelles issues de la base de données USPS. Nous effectuons deux ensembles d'expériences. Dans le premier, une comparaison des performances des noyaux est réalisée après la sélection manuelle des valeurs des hyper-paramètres. Le noyau KMOD donne le plus faible taux d'erreur. Aussi, la complexité des classifieurs produits est moindre. Dans le deuxième ensemble d'expériences, nous effectuons une optimisation automatique des paramètres de noyaux. Les résultats sont montrés pour plusieurs types de noyaux et quelques schémas de combinaison des classifieurs. L'optimisation automatique permet de choisir des modèles compactes avec le moins de vecteurs de support. Aussi, elle permet d'assigner des valeurs individuelles propres à chaque noyau de l'ensemble de classifieurs. Ce degré de liberté permet d'améliorer les performances du système.

Idem pour les chiffres indiens, nous effectuons une optimisation automatique et manuelle des hyper-paramètres. Les meilleurs noyaux sont le RBF et KMOD. L'efficacité de la sélection de modèle automatique est démontrée pour les approches locale et globale.

CHAPITRE 9

CONCLUSION GÉNÉRALE

Le SVM est l'un des modèles de classification à avoir le plus marqué la reconnaissance de formes en fournissant un cadre théorique statistique et non plus connexioniste à la théorie de l'apprentissage. Ce classifieur est à même de prédire des frontières de décision complexes grâce à l'introduction de noyaux non linéaires (dits de Mercer) [1, 18]. Ces noyaux, de par leur variété, constituent l'une des avenues les plus prometteuses de la théorie des SVMs. Les noyaux de Mercer, jumelés au SVM, permettent de transformer les données dans un espace de travail (dit espace augmenté) de dimension arbitraire. Ensuite, un hyper-plan de séparation qui maximise la marge y'est trouvé. Ce processus de projection des données dans l'espace induit par le noyau est équivalent à une extraction de caractéristiques systématique sur les données d'entrée. Il permet d'augmenter ou de diminuer la capacité du classifieur. La dimension de cet espace augmenté n'est pas connue et est éventuellement infinie pour certains noyaux et certaines valeurs de leurs paramètres. La maximisation de la marge dans le SVM est une procédure de sélection de modèle implicite à l'apprentissage qui permet de minimiser l'erreur d'apprentissage tout en réduisant la complexité du classifieur. Malheureusement, les paramètres de noyaux (hyper-paramètres) ne sont pas pris en compte pendant l'apprentissage. N'ayant pas d'information a priori sur le problème, le classifieur entraîné est loin des performances escomptées. Le cas échéant, la capacité du classifieur n'est pas adaptée aux données considérées.

Par ailleurs, le SVM est un modèle dont la complexité est fonction du nombre de vecteurs de support. Au plus, ce classifieur contient autant de paramètres que de vecteurs d'apprentissage. Contrairement au PMC, la complexité du SVM ne dépend pas du nombre de caractéristiques. Elle dépend seulement de la complexité du problème et des valeurs des hyper-paramètres. Cette propriété rend le classifieur plus robuste au phénomène du sur-apprentissage lorsque la quantité de données disponible est modeste ou non suffisante.

Dans le cadre de cette thèse, nous avons étudié le problème de l'optimisation des hyper-paramètres du SVM. En particulier, nous nous sommes intéressés à l'optimisation des paramètres du noyau par la sélection automatique du meilleur modèle. Dans ce sens, nous proposons une nouvelle méthodologie basée sur la minimisation automatique d'un estimé empirique de l'erreur de généralisation. Cette dernière est une approximation de la probabilité d'erreur estimée sur un ensemble de données indépendant de celui de l'apprentissage (Chapitre 5). Pour des fins de comparaison, nous avons aussi considéré deux autres critères algébriques que sont la dimension VC et l'Erreur de Validation Croisée Généralisée GACV.

La dimension VC est une expression dérivée des travaux sur le risque structurel dont dérive le SVM. Elle est utilisable uniquement dans le cas du SVM de type L2. En outre, c'est une borne très conservatrice de l'erreur de généralisation. Les expérimentations réalisées dans le chapitre 7 montrent que le critère, bien que minimisé, ne garantit pas une réduction de l'erreur. En particulier, nous avons constaté que lorsqu'un sur-apprentissage initial apparent des données est présent, ce critère est inefficace. Aussi, à la convergence, ce critère produit énormément de vecteurs de support.

Par ailleurs, le GACV est un critère statistique approximant l'erreur de validation croisée (LOO) [124, 125, 123, 122]. Il est estimé sur les données d'apprentissage et ne prend en compte que les vecteurs de support et leurs paramètres. Nous avons alors développé un algorithme de minimisation automatique du GACV pour les variantes L1 et L2 du SVM. Les expériences effectuées dans le chapitre 7 montrent que la minimisation du GACV n'est effective que lorsque la configuration initiale du classifieur correspond à une capacité réduite ou moyenne. Le cas échéant, on observe une réduction de l'erreur à la convergence. Ceci s'accompagne d'une réduction du nombre de vecteurs de support pour GACV L1 (et pour GACV L2 lorsque C est très grand). Pour un sur-apprentissage initial du SVM (grande capacité), aussi bien la dimension VC que le GACV L1 et le GACV L2 se situent sur un plateau de la fonction objective et les critères finissent à leurs valeurs initiales.

Le critère de l'erreur empirique est une estimation de la probabilité d'erreur des données de validation. Nous avons proposé un nouvel algorithme qui minimise la probabilité d'erreur en corrigeant progressivement les paramètres du noyau. L'algorithme prend en compte tout type de noyau. Sur des données biclasses, la méthode montre une grande efficacité à réduire l'erreur de test à la convergence. Aussi, nous démontrons que la procédure réduit grandement le nombre de vecteurs de support.

Le calcul de la probabilité de l'erreur est basé sur l'estimation de la probabilité à posteriori des données. Or, la sortie du SVM fournit seulement une valeur brute qui de surcroît n'est pas normalisée. Nous avons alors utilisé une procédure de calibrage des sorties du SVM qui consiste à adapter une fonction logistique à la sortie du classifieur. Les paramètres de celle-ci sont estimés en minimisant l'entropie croisée des données de validation.

La modification des paramètres du noyau pendant l'optimisation est basée sur l'estimation du gradient du critère par rapport à ces paramètres. Ceci représente une analyse de sensibilité qui se fait en deux étapes :

- estimation du gradient de l'erreur empirique par rapport aux paramètres α_i du SVM,
- estimation du gradient des paramètres α_i par rapport aux hyper-paramètres.

La valeur du paramètre de compromis C est aussi un hyper-paramètre qui régit la performance du SVM. Ce paramètre sert à fixer le compromis entre la minimisation de l'erreur d'apprentissage et la maximisation de la marge. En pratique, le comportement du SVM est sensible à la valeur de C uniquement si les données d'apprentissage ne sont pas séparables. Dans ce cas, il existe des valeurs critiques qui peuvent compromettre la performance du classifieur. Une très grande valeur de C (quelques milliers) peut faire que la fonction objective minimisée par le SVM ne soit plus convexe et empêcherait sa convergence. Une très faible valeur de C, tend à diminuer la capacité du classifieur. L'influence des noyaux est dans ce cas altérée et la frontière de décision produite tend vers une droite.

Des valeurs intermédiaires entre ces deux valeurs extrêmes, permettent de choisir des valeurs adéquates des paramètres du noyau. Toutefois, l'optimisation automatique de la valeur de C n'est pas possible pour le SVM L1 parce que les paramètres α_i ne sont pas toujours fonctions continues de C. Donc, leurs dérivées par rapport à C peuvent ne pas exister. Pour le SVM L2, cette optimisation est possible en intégrant ce paramètre dans la matrice de Grahm-Schmidt [24]. Cependant ce modèle produit une solution plus complexe que celle du SVM L1 avec beaucoup de vecteurs de support.

Dans notre système, nous avons opté pour le modèle SVM L1 pour lequel l'adaptation de C est manuelle. Du point de vue pratique, l'apprentissage d'un ensemble de SVMs pour la classification de chiffres manuscrits dans l'approche un-contre-un, n'est pas très sensible à la valeur de C comme nous l'avons remarqué dans le chapitre 8. Ceci n'est toujours pas le cas dans l'approche un-contre-tous où le problème est potentiellement non séparable.

Pour des données multiclasses, nous avons développé deux méthodologies d'optimisation des hyper-paramètres. La première désignée *«approche locale»*, minimise les erreurs empiriques des SVMs pris individuellement. La deuxième, désignée *«approche globale»*, minimise l'erreur empirique multiclasse en optimisant l'ensemble des classifieurs simultanément.

Sur le plan expérimental, nous avons procédé en deux étapes. Dans la première, nous établissons l'efficacité de la méthodologie proposée pour l'optimisation d'un seul SVM. Nous avons alors considéré un problème à deux classes dont les données ont été générées artificiellement. Ensuite, nous avons étudié le comportement des procédures d'optimisation pour différentes configurations initiales du classifieur. Aussi, nous avons étudié l'influence de la taille de l'ensemble de validation sur la procédure.

Dans le chapitre 8, l'expérimentation sur des données multiclasses est effectuée. Celle-ci est scindée en trois parties. Dans la première, nous comparons les approches locale et globale décrites dans le chapitre 6. Dans la deuxième, nous présentons les résultats d'ex-

périence sur la base de données USPS. Ceux-la sont départis en deux. En premier, les performances du système sont calculées en choisissant de façon manuelle les valeurs des hyper-paramètres. Ensuite, nous montrons les résultats de l'optimisation automatique selon plusieurs noyaux et schémas de vote. Dans la dernière partie des expériences, nous exécutons la procédures sur la base de données de chiffres indiens de CENPARMI. Idem, nous effectuons une procédure de sélection manuelle des paramètres que nous comparons ensuite avec l'optimisation automatique. Sur les deux bases de données, nous montrons que la méthodologie permet de minimiser l'erreur de généralisation et le nombre de vecteurs de support.

9.1 Contributions majeures

L'essence de cette thèse découle du besoin de comparer les performances des différents noyaux de SVM en classification. En particulier, la nécessité de pouvoir choisir de façon automatique les paramètres du noyau s'est imposée lorsque nous avons introduit notre nouveau noyau KMOD dans [5, 8] (Chapitre 3, section 3.4). Sur un benchmark de UCI et sur USPS, ce noyau est meilleur que le RBF (Tableaux IV, XIX et XXI).

Nous avons proposé une nouvelle méthodologie de sélection de modèle basée sur l'estimation et la minimisation d'un estimé de la probabilité d'erreur. Celui-ci est calculé sur des données réelles de la même distribution que les données du problème. La méthode est originale et intègre la sélection de modèle dans le processus d'apprentissage de façon transparente. Nous démontrons empiriquement que la probabilité d'erreur minimise la complexité du SVM.

Aussi, nous avons proposé dans le chapitre 5 (section 5.4) un nouvel algorithme de minimisation du critère GACV pour les variantes L1 et L2 du SVM. Pour des fin de comparaison, nous avons développé la procédure de minimisation de la dimension VC également.

Sur des données multiclasses, nous proposons deux méthodologies de sélection de modèle basées sur l'optimisation individuelle des SVM (approche locale) et sur l'optimisation simultanée des SVM (approche globale). Les expériences réalisées dans le chapitre 8 montrent que les deux méthodes diminuent l'erreur et la complexité à travers les $K(K-1)/2$ SVM de l'approche un-contre-un.

9.2 Perspectives

- *Estimation du gradient* : L'estimation du gradient de l'erreur empirique par rapport aux hyper-paramètres est basée sur l'équation 5.20 (Chapitre 5). Cette équation suppose que les données sont séparables. C'est pourquoi nous avons choisi d'utiliser le schéma de décomposition en un-contre-un où les sous-problèmes sont plus simples, et donc vraisemblablement séparables. Il serait alors intéressant de modifier l'équation de façon à tenir compte des vecteurs de support situés à l'intérieur et au delà de la marge. Ceci permet de gérer plus efficacement les cas non séparables. La procédure pourrait alors s'accommoder facilement avec l'approche un-contre-tous (Chapitre 3)
- *Approche globale* : L'approche globale d'optimisation est efficace mais assez prohibitive en temps de calcul. En effet, à chaque estimation de la correction à apporter aux hyper-paramètres du classifieur (i, j), nous considérons l'ensemble des données des classes (les classes i et j plus les classes marginales). Une modification de l'algorithme qui permet de simplifier ce calcul rendrait l'approche globale plus rapide est plus intéressante.
- *Sélection de modèle en ligne* : La méthodologie de sélection de modèle proposée est basée sur l'estimation de l'erreur empirique. Cette dernière est calculée à la fin de chaque procédure d'apprentissage. Une avenue très intéressante serait de pouvoir optimiser les hyper-paramètres du SVM au cours de l'apprentissage de ses paramètres α_i. Ceci permettrait d'accélérer grandement le calibrage du classifieur et de l'intégrer dans l'apprentissage.

– *Détermination automatique de pertinence* : La puissance du SVM réside dans la possibilité de choisir des noyaux de natures différentes pour inférer des frontières de décision de complexité arbitraire. Le paramètre σ du RBF par exemple, permet de fournir une mesure de similarité $k(x_i, x_j)$ connaissant la distance euclidienne $\|x_i - x_j\|^2$ entre les points x_i et x_j de l'espace d'entrée. Ce paramètre traite équitablement les dimensions de l'espace d'entrée. Si toutefois nous assignons un paramètre d'échelle par dimension, il est possible de pondérer les différentes caractéristiques des données en effectuant une optimisation automatique de ces paramètres. Cette procédure est connue sous le nom «Automatic Relevance Determination-ARD».

«Everything should be made as simple as possible, but not simpler»,
Albert Einstein

BIBLIOGRAPHIE

[1] M.A. Aizerman, F.M. Braverman, and L.I. Rozoner. The probability problem in pattern recognition learning and the method of potential functions. *Automation and Remote Control*, 25(9) :1175–1190, 1964.

[2] H. Akaike. Fitting autoregressive models for prediction. *Annals of the Institute of Statistical Mathematics*, 21 :243–247, 1969.

[3] H. Akaike. Information theory and an extension of the maximum likelihood principle. In B.N. Petrov and Csaki, editors, *proceedings of the 2nd International Symposium on Information Theory*, pages 267–281, Tsahkadsov, Armenia, USSR, 1973.

[4] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary : A unifying approach for margin classifiers. In *Proc. 17th International Conf. on Machine Learning*, pages 9–16. Morgan Kaufmann, San Francisco, CA, 2000.

[5] N.E. Ayat, M. Cheriet, L. Remaki, and C.Y. Suen. Kmod- a new support vector machine kernel for pattern recognition. application to digit image recognition. In *Proceedings of the IEEE International Conference on Document Analysis and Recognition*, pages 1215–1219, Seattle,USA, Sept. 2001.

[6] N.E. Ayat, M. Cheriet, and C.Y. Suen. Un système neuro-flou pour la reconnaissance de montants numériques de chèques arabes. In *Colloque International Francophone sur l'Écrit et le Document*, pages 171–180, Lyon,France, Jul. 2000.

[7] N.E. Ayat, M. Cheriet, and C.Y. Suen. Empirical error based optimization of svm kernels. application to digit image recognition. In *Proceedings of the International Workshop on Handwriting Recognition*, Niagara on the lake, Canada, 2002.

[8] N.E. Ayat, M. Cheriet, and C.Y. Suen. Kmod-a two parameter svm kernel for pattern recognition. In *Proceedings of the IEEE International Conference on Pattern Recogntion*, Quebec, Canada, 2002.

[9] N.E. Ayat, M. Cheriet, and C.Y. Suen. Optimization of the svm kernels using an empirical error minimization scheme. In S.W. Lee and A. Verri, editors, *Pattern Recognition with Support Vector Machines*, volume 2388 of *LNCS*, pages 354–369. Berlin Heidelberg, July 2002.

[10] G.J. Balm. An introduction to optical character reader considerations. *Pattern*

Recognition, 2(3) :151–166, 1970.

[11] A.R. Barron. Predicted squared error : A criterion for automatic model selection. In S.J. Farlow, editor, *SelfOrganizing Methods in Modeling*, volume 54, chapter 4, pages 87–103. Marcel Dekker, New York, 1984.

[12] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10) :2385–2404, 2000.

[13] Y. Bengio. Gradient-based optimization of hyper-parameters. *Neural Computation*, 12(8) :1889–1900, 2000.

[14] D. P. Bertsekas. Nonlinear programming, 1999.

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

[16] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[17] V. Blanz, B. Scholkopf, H.H. Bulthoff, C. Burges, V. Vapnik, and T. Vetter. Comparison of view-based object recognition algorithms using realistic 3d models. In *ICANN*, pages 251–256, 1996.

[18] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, Pittsburg, 1992.

[19] L. Bottou and V.N. Vapnik. Local learning algorithms. *Neural Computation*, 4(6) :888–900, 1992.

[20] J. Bromley and E. Sackinger. Neural-network and k-nearest-neighbor classifiers. Technical Report 11359-910819-16TM, ATT, 1991.

[21] C. Burges. A tutorial on support vector machine for pattern recognition, 1998.

[22] Chris J.C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector machines. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 375. The MIT Press, 1997.

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

[24] O. Chapelle and V. Vapnik. Choosing multiple parameters for support vector ma-

chines. *Advances in Neural Information Processing Systems*, 03(5), March 2001.

[25] M. Cheriet and C.Y. Suen. Extraction of key letters for cursive script recognition. pattern recognition letters 14 (1993), pp. 1009-1017.

[26] Y.C. Chim, A.A. Kassim, and Y. Ibrahim. Dual classifier system for handprinted alphanumeric character recognition. pattern analysis and applications (1998) 1 :pp. 155-162.

[27] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3) :273–297, 1995.

[28] Courant and R. Hilbert. *Methods of Mathematical Physics*. Interscience, 1953.

[29] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2 :263–286, 1995.

[30] H. Drucker, R. Schapire, and P.Y. Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4) :705–719, 1993.

[31] C.E. Dunn and P.S.P. Wang. Character segmentation techniques for handwritten text : a survey. In *Proceedings of the IEEE Conference on Pattern Recognition*, volume 2, pages 577–580, Hague.

[32] R.A. Fisher. The use of multiple measurements in taxonomic problems, 1936.

[33] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1) :1–58, 1992.

[34] Gill, Murray, and Wright. Practical optimization, 1981.

[35] M. Gilloux. Research into the new generation of character and mailing address recognition systems at the french post office research center. *Pattern recognition letters*, 14(4) :267–276, 1993.

[36] F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. Technical Report 1599, Massachusetts Institute of Techology, MA, USA, 1996.

[37] Federico Girosi. An equivalence between sparse approximation and support vector machines. Technical Report AIM-1606, 1997.

[38] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. Unipen project of on-line data exchange and benchmarks. In *Proceedings of the IEEE Conference on Pattern Recognition*, pages 29–33, Jerusalem. IEEE Computer Society Press.

[39] S. J. Hanson and D. J. Burr. Minkowski-r back-propagation : Learning in connectionist models with non-euclidian error signals. In *Neural Information Processing Systems*.

[40] L.D. Harmon. Automatic recognition of print and script. In *Proc. of the IEEE*, volume 60, pages 1165–1177. IEEE Computer Press.

[41] Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

[42] L. Heutte, T. Paquet, J.V. Moreau, Y. Lecourtier, and C. Olivier. A structural/statistical feature based vector for handwritten character recognition. pattern recognition letters 19 (1998), pp. 629-641.

[43] G.E. Hinton. Learning translation invariant recognition in a massively parallel network. In J.W. de Bakker, A.J. Nijman, and P.C. Treleaven, editors, *Proceedings of PARLE Conference on Parallel Architectures and Languages Europe*, pages 1–13, Berlin, 1987. Springer Verlag.

[44] C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13 :415–425, 2002.

[45] T.S. Jaakola and D. Haussler. Probabilitic kernel regression models. In *Conference on AI and statitics*.

[46] T. Joachims. Making large-scale svm learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11. 1999.

[47] T. Joachims. *The maximum- margin approach to learning text classifiers : method, theory and algorithms*. PhD thesis, University of Dortmund, 2000.

[48] Thorsten Joachims. Text categorization with support vector machines : learning with many relevant features. In *Proc. 10th European Conference on Machine Learning ECML-98*, pages 137–142, 1998.

[49] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *Proc. 16th International Conference on Machine Learning*, pages 200–209. Morgan Kaufmann, San Francisco, CA, 1999.

[50] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy. Improvements to platt's smo algorithm for svm classifier design, 1999.

[51] A. Kolmogorov. Three approaches to the quantitative definition of information. *Problems Inform. Transmission*, 1(1) :1–7, 1965.

[52] Jorma Laaksonen. *Subspace Classifiers in Recognition of Handwritten Digits.* Computer science, Helsinky University of Technology, Finland, May 1997.

[53] J. Larsen, C. Svarer, L.N. Andersen, and L.K. Hansen. Adaptive regularization in neural network modeling. In *Neural Networks : Tricks of the Trade*, pages 113–132, 1996.

[54] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4) :541–551, Winter 1989.

[55] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.

[56] S.W. Lee and A. Verri. *Pattern Recognition with Support Vector Machines*, volume 2388 of *LNCS*. Springer Verlag, Berlin, 2002.

[57] Z. Li, S. Tang, and S. Yan. *Pattern Recognition with Support Vector Machines*, volume 2388 of *Lecture Notes in Computer Science*, chapter Multi-Class SVM Classifier Based on Pairwise Coupling, pages 321–333. Springer Verlag, Berlin Heidelberg, July 2002.

[58] H. Lin, C. Lin, and R. C. Weng. A note on platt's probabilistic outputs for support vector machines. Technical report, National Taiwan University, Taipei 116, Taiwan, 2003.

[59] N. Lindgren. Machine recognition of human language : Part iii-cursive script recognition. *IEEE Spectrum*, 2(5) :104–116, 1965.

[60] S. Loncaric. A survey of shape analysis techniques. pattern recognition (1998), vol.31, no. 8, pp. 983-1001.

[61] D. MacKay. *Bayesian Methods for Adaptive Models*. PhD thesis, 1992.

[62] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3) :415–447, 1992.

[63] D. J. C. MacKay. A practical Bayesian framework for backprop networks. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 839–846, 1992.

[64] C.L. Mallows. Some comments on cp. *Technometrics*, 15(4) :661–675, 1973.

[65] J. Mantas. An overview of character recognition methodologies. *Pattern Recognition*, 19(6) :425–430, 1986.

[66] S. Mika, A. Smola, and B. Scholkopf. An improved training algorithm for kernel fisher discriminants, 2001.

[67] Sebastian Mika, Gunnar Rätsch, and Klaus-Robert Müller. A mathematical programming approach to the kernel fisher algorithm. In *NIPS*, pages 591–597, 2000.

[68] Sebastian Mika, Gunnar Rätsch, Jason Weston, Bernhard Schölkopf, and Klaus-Robert Müller. Fisher discriminant analysis with kernels, 1999.

[69] J. Moody. The effective number of parameters : An analysis of generalization and regularization in nonlinear learning systems. In J. Moody, S. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 847–854. Morgan Kaufmann, San Mateo, CA, 1992.

[70] Miguel Moreira and Eddy Mayoraz. Improved pairwise coupling classification with correcting classifiers. IDIAP-RR 09, IDIAP, 1997. Proceedings of the 10th European Conference on Machine Learning, 1998.

[71] Il-Seok Oh and Ching Y. Suen. Distance features for neural network-based recognition of handwritten characters. ijdar 1998, pp. 73-88.

[72] M. Opper and O. Winther. Gaussian processes and svm : Mean field and leave-one-out. In P. Bartlett, C. Burges, and D. Shuurmans, editors, *Advances in large margin classifiers*, pages 311–326. 2000.

[73] E. Osuna, R. Freund, and F. Girosi. Improved training algorithm for support vector machines, 1997.

[74] E. Osuna, R. Freund, and F. Girosi. Training support vector machines :an application to face detection, 1997.

[75] T. Pavlidis and F. Ali. Computer recognition of handwritten numerals by polygonal approximations. *IEEE Trans. Systems, Man, and Cybernetics*, 5(6) :610–614, November 1975.

[76] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3), October 1999.

[77] J.C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 12. 1999.

[78] T. Poggio. On optimal nonlinear associative recall. *Biological Cybernetics*, 19 :201–209, 1975.

[79] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317(26) :314–319, 1985.

[80] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C, the art of scientific computing.* Cambridge University Press, second edition, 1992.

[81] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for adaboost. *Machine Learning*, 43(3) :287–320, 2001.

[82] J. Rissanen. Modeling by shortest data description. *Automatica*, 14 :465–471, 1978.

[83] F. Rosenblatt. *Principles of Neurodynamics.* Spartan Books, New York, 1962.

[84] V. Roth and V. Steinhage. Nonlinear discriminant analysis using kernel functions technical report nr, 1999.

[85] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error backpropagation. In J. L. McClelland In D. E. Rumelhart and the PDP research group, editors, *Parallel distributed processing : explorations in the microstructure of cognition*, pages 318–362. 1986.

[86] M. Schmidt. Identifying speakers with support vector networks, 1996.

[87] B. Scholkopf. *Support vector learning.* PhD thesis, Universität Berlin, Berlin, Germany, 1997.

[88] B. Scholkopf, C. Burges, and A. Smola. Introduction to support vector learning. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 1. 1999.

[89] B. Scholkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In *First International Conference on Knowledge Discovery and Data Mining.*

[90] B. Scholkopf, C. Burges, and V. Vapnik. Incorporating invariances in support vector learning machines. In C. Von der Malsburg, W. von Seelen, J. C. Vorbruggen, and B. Sendhoff, editors, *Lecture notes in computer science*, pages 47–52. 1996.

[91] B. Scholkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson. Estimating the support of a high-dimensional distribution, 1999.

[92] B. Scholkopf, P. Simard, A. Smola, and V. Vapnik. Prior knowledge in support vector kernels. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems*. 1998.

[93] B. Scholkopf, O. Smola, R. Williamson, and P. Bartlett. New support vector algorithms. *Neural Computation*, 12 :1083–1121, 2000.

[94] S. Sigurdsson, J. Larsen, and L.K. Hansen. On comparison of adaptive regularization methods. In K. Paliwa T. Adali J. Larsen E. Wilson S. Douglas B. Widrow, L. Guan, editor, *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, volume 11, pages 221–230, 2000.

[95] P. Simard, Y. LeCun, and J. Denker. Efficient pattern recognition using a new transformation distance. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems*, pages 50–58. 1993.

[96] A. Smola. *Learning with Kernels*. PhD thesis, GMD First, Berlin, Germany, 1998.

[97] A. Smola, O. Mangasarian, and B. Scholkopf. Sparse kernel feature analysis, 1999.

[98] A. Smola and B. Scholkopf. A tutorial on support vector regression, 1998.

[99] Alex J. Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proc. 17th International Conf. on Machine Learning*, pages 911–918. Morgan Kaufmann, San Francisco, CA, 2000.

[100] P. Sollich. Bayesian methods for support vector machines : Evidence and predictive class probabilities. *Machine Learning*, 46(1/3) :21, 2002.

[101] S.N. Srihari. Recognition of handwritten and machine-printed text for postal address interpretation. *Pattern Recognition Letters*, 14(4) :291–302, 1993.

[102] M.O. Stitson, A. Gammerman, V. Vapnik, V.Vovk, C. Watkins, and J. Weston. Support vector anova decomposition, 1997.

[103] N.W. Strathy. Master thesis (1993), concordia university, montreal.

[104] C.Y. Suen, M. Berthod, and S. Mori. Automatic recognition of hand printed characters – the state of the art. In IEEE Computer Press, editor, *Proceedings of the IEEE*, volume 68, pages 469–487, 1980.

[105] C.Y. Suen, R. Legaut, C. Nadal, M. Cheriet, and L. Lam. Building a new generation of handwriting recognition systems. *Pattern Recognition Letters*, 14 :303–315, April 1993.

[106] C.Y. Suen, C. Nadal, R. Legault, T.A. Mai, and L. Lam. Computer recognition of unconstrained handwritten numeral. In IEEE Computer Press, editor, *Proceedings of the IEEE*, volume 80, pages 1162–1180, 1992.

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

[108] C.C. Tappert, C.Y. Suen, and T. Wakahara. The state of the art in online handwriting recognition. *IEEE Trans. Pattern Analysis Machine Intelligence*, 12(8) :787–808, 1990.

[109] D. Tax and R. Duin. Data domain description by support vectors, 1999.

[110] D. Tax and R. Duin. Support vector data description. *Pattern Recognition Letters*, 20(11-13) :1191–1199, December 1999.

[111] A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill-Posed Problems*. V.H. Winston and Sons, Washington, DC, 1977.

[112] Michael E. Tipping. Sparse kernel principal component analysis. In *NIPS*, pages 633–639, 2000.

[113] O.D. Trier, A.K. Jain, and T. Taxt. Feature extraction methods for character recognition : A survey. *Pattern Recognition*, 29(4) :641–662, 1996.

[114] R. J. Vanderbei. Interior point methods : Algorithms and formulations. *ORSA Journal of Computing*, 6(1) :32–34, 1994.

[115] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, NY, USA, 1995.

[116] V. Vapnik. *Statistical Learning Theory*. NY, USA, 1998.

[117] V. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5), September 1999.

[118] Vladimir Vapnik and Olivier Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12(9) :2013–2036, 2000.

[119] V.N. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian.* Nauka, Moscow, Russia, 1979.

[120] V.N. Vapnik. *Estimation of Dependences Based on Empirical Data.* springer Verlag, Berlin, 1982.

[121] P. Vincent and Y. Bengio. Kernel matching pursuit. *Machine Learning*, 48 :165–187, 2002.

[122] G. Wahba. Support vector machines, reproducing kernel hilbert spaces, and randomized gacv, 1998.

[123] G. Wahba. Support vector machines, reproducing kernel hilbert spaces, and randomized gacv, 1998.

[124] G. Wahba. The bias-variance trade-off and the randomized gacv. *Advances in Neural Information Processing Systems*, 11(5), November 1999.

[125] Grace Wahba, Yuedong Wang, Chong Gu, Ronald Klein, and Barbara Klein. Structured machine learning for "soft" classification with smoothing spline ANOVA and stacked tuning, testing, and evaluation. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 415–422. Morgan Kaufmann Publishers, Inc., 1994.

[126] D.S. Watkins. *Fundamentals of Matrix Computations.* Wiley, New York, 1991.

[127] P. J. Werbos. *Beyond regression : New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Masters Thesis, Harvard University, 1974.

[128] J. Weston, A. Gammerman, M. Stitson, V. Vapnik, V. Vovk, and C. Watkins. Density estimation using support vector machines, 1997.

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