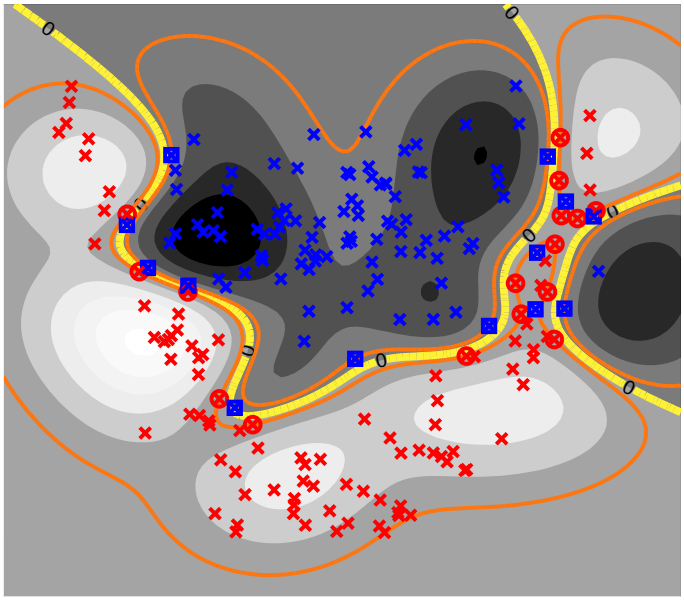
Institut National des Sciences Appliquées



Thèse de doctorat

par **Gaëlle Loosli**

pour obtenir le grade de

Docteur de l'Institut National des Sciences Appliquées de Rouen

Discipline : Informatique

Méthodes à noyaux pour la détection de contexte

Vers un fonctionnement autonome des méthodes à noyaux

Jury :

Florence d'Alche-Buc	Professeur à l'Université d'Evry-Val d'Esonne
Léon Bottou (Rapporteur)	Chercheur à NEC, Princeton, Etats-Unis
Stéphane Canu (Directeur)	Professeur à l'INSA de Rouen
Antoine Cornuéjols	Professeur à l'INAPG, Paris
Bernadette Dorizzi (Rapporteur)	Professeur à l'INT, Evry
Alain Rakotomamonjy	Professeur à l'Université de Rouen

Laboratoire d'Informatique, de Traitement de l'Information et des Systèmes (LITIS) - INSA de Rouen
Avenue de l'Université, 76801 Saint Etienne du Rouvray Cedex

Table des matières

Remerciements **vii**

Introduction **ix**

Notations **xix**

I Position des problèmes abordés **1**

1 Contexte : apports et défis pour l'apprentissage **3**

1.1 Apprentissage et émotions 4

1.2 État de l'art et travaux connexes 5

1.2.1 Mise en perspective succincte de la théorie statistique de l'apprentissage 6

1.2.2 Apprentissage pour la perception d'états affectifs 9

1.3 Ethique 12

1.4 Proposition de cadre de travail 12

1.4.1 Architecture pour la détection de contexte humain 12

1.4.2 Acquisition des données réelles 14

2 Apprentissage automatique et optimisation quadratique **17**

2.1 L'apprentissage, aspects théoriques 18

2.1.1 Cadre général 18

2.1.2 Cadre fonctionnel 20

2.1.3 Coût et Régularisation 21

2.1.4 Sur les objectifs d'un apprentissage réaliste 21

2.2 L'optimisation quadratique sous contraintes pour l'apprentissage 22

2.2.1 SVM : un problème quadratique sous contraintes 22

2.2.2 La propriété utilisée : la parcimonie 23

2.2.3 Les outils utilisés 23

2.2.4 Structure commune des algorithmes de résolution 23

2.2.5 Méthodes de décomposition 24

2.2.6 Résolution d'un problème quadratique 25

2.3 Méthodes alternatives non optimales 26

2.3.1 CVM . . . 26
2.3.2 LASVM . . . 28
2.4 Résumé des méthodes de résolution . . . 29

II Outils et contributions 33

3 SimpleSVM 35
3.1 Relation entre primal et dual . . . 35
3.2 Structure et détails de SimpleSVM . . . 37
3.2.1 Activation d'une contrainte : projection . . . 38
3.2.2 Désactivation d'une contrainte . . . 39
3.2.3 Convergence . . . 39
3.3 Variations sur le même thème . . . 40
3.3.1 Le SVM à une classe : OC-SVM . . . 40
3.3.2 Le ν-SVM . . . 42
3.3.3 Reprise à chaud de l'algorithme . . . 43

4 Réglage des hyper-paramètres grâce au chemin de régularisation 47
4.1 Le chemin de régularisation . . . 48
4.1.1 Un chemin linéaire par morceaux . . . 49
4.1.2 Limitations . . . 49
4.2 Le chemin à contresens pour le ν-SVM . . . 50
4.2.1 Points de I_w et détection de $outC$ et $out0$ . . . 50
4.2.2 Points de I_C et I_0 et détection de inC et $in0$ . . . 51
4.2.3 Algorithme du chemin de régularisation . . . 52
4.3 Évaluation des performances le long du chemin et arrêt anticipé . . . 52
4.4 Résultats expérimentaux . . . 53
4.4.1 Illustration des chemins . . . 53
4.4.2 Efficacité du chemin de régularisation du ν-SVM . . . 53
4.4.3 Sur le critère d'arrêt . . . 54

III Applications et mise en œuvre 57

5 En pratique : boîte à outils SimpleSVM 59
5.1 Une boite à outils en Matlab . . . 59
5.1.1 Initialisation . . . 60
5.1.2 Résolution du système linéaire . . . 60
5.1.3 Résolution Dynamique . . . 61
5.1.4 Ajout d'un point dans la solution . . . 61

5.1.5 Critère d'arrêt . . . 61
5.1.6 Cache . . . 61
5.1.7 Fonctionnement général . . . 62
5.1.8 Description des structures et des principales fonctions . . . 62
5.2 Étude des critères d'arrêt et comparaison des méthodes de résolution . . . 64
5.2.1 Notations et définitions . . . 64
5.2.2 C-SVM (SimpleSVM et SMO) . . . 64
5.2.3 ν-SVM . . . 65
5.2.4 CVM . . . 65
5.2.5 Résultats expérimentaux . . . 66

6 Invariances et très grandes bases de données 71
6.1 De l'influence des invariances . . . 72
6.2 De la formulation des invariances . . . 73
6.3 Formulation unifiée . . . 74
6.3.1 Agrandissement artificiel de la base d'apprentissage . . . 74
6.3.2 Adaptation de la distance aux invariances . . . 74
6.3.3 Approximations polynomiales . . . 75
6.4 Des invariances dans la pratique pour la reconnaissance de caractères . . . 75
6.4.1 Vecteurs tangents . . . 75
6.4.2 Champs de déformation . . . 76
6.5 Traitement par méthode hors ligne - SimpleSVM . . . 79
6.5.1 SimpleSVM Invariant . . . 79
6.5.2 Application à la reconnaissance de caractères . . . 79
6.5.3 Discussion . . . 80
6.6 Traitement par méthode en ligne - LASVM . . . 80
6.6.1 Du réglage des déformations . . . 80
6.6.2 Du mode de sélection . . . 81
6.6.3 Des effets de l'optimisation complète . . . 82
6.6.4 Du nombre de reprocess . . . 82
6.6.5 Des résultats globaux de LASVM pour les invariances sur MNIST . . . 83

7 Détection de changement 85
7.1 Familles exponentielles non paramétriques et méthodes à noyaux . . . 86
7.2 Tests Statistiques et détection de rupture . . . 86
7.2.1 Tests sur la variance des sorties de l'OC-SVM . . . 87
7.2.2 Tests sur le taux de mauvaise classification . . . 88
7.2.3 Tests sur la somme cumulée des sorties de l'OC-SVM . . . 88
7.3 Expérimentations . . . 89

7.3.1 Illustration de la méthode sur des données synthétiques 89
7.3.2 Illustration sur signaux réels 89
7.3.3 Acquisition de signaux 90
7.3.4 Utilisateur monitoré dans un contexte anodin 90
7.3.5 Utilisateur monitoré dans le cadre d'un jeu vidéo 92

Conclusions et perspectives **95**

Annexe 1 - Capteurs **107**

Annexe 2 - Bases de données **111**

Remerciements

« Je dédie cette thèse
Au petit être qui grandit en moi. »

Gaëlle

Je tiens à remercier en premier lieu les membres du jury, Florence d'Alche, Léon Bottou, Antoine Cornuéjols, Bernadette Dorizzi et Alain Rakotomamonjy pour avoir pris le temps de s'intéresser à mes travaux.

Je remercie Stéphane Canu, directeur de thèse et encadrant mais aussi ami, qui m'a beaucoup appris et beaucoup apporté, autant sur le plan scientifique que personnel.

Je remercie les collègues et amis de l'INSA et du LITIS pour leur accueil et pour certains, pour leurs enseignements qui m'ont menée à faire une thèse, en particulier Alain, Nico, Nico, Gilles, Florence et Brigitte.

Je remercie également *le groupe des thésards et associés*, les joueurs de XBlast, les buveurs de thé, les anciens qui sont devenus docteur, les nouveaux qui ne connaissent pas XBlast et les associés qui pour une raison ou pour une autre supportent les conversations geek des thésards, pour l'atmosphère ludique et l'entraide qu'ils apportent. Merci donc à Vincent, Fabien, Frédéric, Karina, Olivier, Sam, Sarah, Karine, Keyang, Emilie et tous les autres. Il me faut ajouter quelques mots pour Fabien pour avoir servi de cobaye au cours de mes expériences. J'en profite pour mentionner notre article commun qui a quelques difficultés à s'insérer dans le reste de cette thèse [Delorme and Loosli, 2006].

J'ai eu au cours des ces trois dernières années beaucoup d'occasions de voyager. Je remercie Alex et Vishy pour leur accueil à Canberra ainsi que Léon et Ronan pour leur accueil à Princeton. Travailler dans ces laboratoires a certainement beaucoup contribué à l'orientation de mes recherches et des mes centres d'intérêt.

J'ai une pensée particulière pour mes parents, mes frères et soeur, Yann, Cédric, Clémentine, qui n'ont jamais vraiment bien su ce que je faisais et dont je me suis souvent éloignée lors de mes différents voyages.

J'ai ensuite plusieurs mentions spéciales pour quelques personnes auxquelles je tiens particulièrement et qui chacune à leur façon m'ont permis d'arriver au bout de mes études. Karine, qui sait me dire avec franchise quand je me trompe et m'ouvrir les yeux, qui a toujours été là à chaque fois que j'ai eu besoin d'aide et qui est aussi là pour partager les meilleurs moments. Alice, qui m'a encouragée à assumer la personne que je suis avec bonne humeur et qui veille sur moi de loin. Adeline, qui entre dans plusieurs catégories de ces remerciements mais qui est en premier lieu une de mes plus ancienne amies.

Et enfin, je te remercie, Pierre-Olivier, de ton soutient et de ta patience, en particulier pour les derniers mois de rédaction.

Introduction

« J'ai trouvé cette phrase dans le courrier électronique de Julia : « Nous n'avons rien à perdre. » Mais ils ont tout perdu - leur société, leur vie, tout. L'ironie de la chose est que leur tentative a été couronnée de succès : l'essaim a résolu le problème qu'ils lui avaient donné à résoudre.
Mais il ne s'est pas arrêté en si bon chemin ; il a continué à évoluer.
Et ils l'ont laissé faire.
Ils n'ont pas compris ce qu'ils faisaient.
Je redoute que cette inscription ne figure un jour sur la pierre tombale de la race humaine.
J'espère qu'il n'en sera rien.
La chance nous sourira peut-être. »

Michael Crichton

DANS SON LIVRE « LA PROIE » , Crichton [2003] met en scène des chercheurs qui parviennent à créer un système autonome, un essaim dont la première vocation est d'être une caméra autonome mignature. Ce système composé de nano-éléments est capable d'apprendre, de s'adapter à son environnement mais aussi de se reproduire. Le scénario joue sur les peurs qui accompagnent les avancées scientifiques en nanotechnologie et en intelligence artificielle. Que se passerait-il si les scientifiques parvenaient en effet à créer un système aussi proche du vivant ? Pourrait-on en perdre le contrôle ? Cependant, d'autres questions se posent à la lecture d'un tel scénario : sommes-nous proches de réaliser un système autonome ? Sans même ajouter la contrainte de la nanotechnologie, existe-t-il des systèmes qui puissent apprendre et s'adapter à leur environnement ?

L'état actuel de nos relations avec les outils technologiques qui nous entourent pose des problèmes. Chacun d'entre nous s'est trouvé un jour devant un appareil récalcitrant, un magnétoscope incompréhensible ou un téléphone portable qui sonne au cinéma alors que pourtant « J'étais sûr d'avoir coupé la sonnerie ». Chaque appareil est livré avec son mode d'emploi de 96 pages et nombre de personnes abandonnent l'idée d'en acheter certains car ils sont persuadés de ne pas savoir les faire fonctionner. A ce titre, la technologie a pris le pas sur l'humain et ce dernier doit s'adapter sans cesse. Un outil plus perfectionné n'est pas une garantie de simplicité d'utilisation, bien au contraire. Prenons l'exemple de la complétion automatique des mots lorsque nous rédigeons un message sur notre téléphone. Le mot proposé est, au mieux, le mot le plus courant correspondant aux lettres possibles, or le contenu du message est bien souvent lié à la situation de celui qui le compose. Ainsi, les mots « voiture » et « toiture » sont composés de la même séquence de touches. Selon que l'utilisateur est dans un magasin de bricolage au rayon *construction* ou qu'il est sur un parking, il est probable que le mot voulu ne soit pas le même. Pourtant, ce sera toujours le mot « voiture » qui sera de préférence proposé.

Cette relation entre technologie et humains semble acquise jusque dans la publicité (« Tu ne sais pas cliquer ! »). Pourtant, il est envisageable de renverser ce rapport et de faire en sorte que ce soit la technologie qui s'adapte à l'humain. Il est nécessaire de repenser les désormais incontournables téléphones portables pour qu'il

n'y ait plus besoin de penser à arrêter la sonnerie (et à la réactiver) et que l'appareil « sache » de lui-même quel est le comportement le plus adapté à la situation. Le téléphone serait alors presque aussi autonome que l'essaim-caméra de M. Crichton.

Dans un futur proche, les objets anodins du quotidiens seront instrumentés et ce grâce aux progrès faits dans la miniaturisation et l'intégration des composants électroniques. Les vêtements intelligents, les cuisines intelligentes, les véhicules intelligents font l'objet de très nombreuses études. S'il est en effet certain que nous sommes capables d'instrumenter aussi bien les voitures que les tee-shirt, il est en revanche moins facile de justifier le terme « intelligent ». Les objets truffés de capteurs ont la capacité de récupérer un grand nombre d'informations brutes sur leur environnement. Cette étape est essentielle pour aboutir à une technologie intelligente. En effet, les capteurs sont aux objets ce que sont les sens aux être vivants. Sans ces intermédiaires, il nous est impossible d'interagir avec notre environnement ou même de le prendre en compte. De la même façon, un téléphone ne peut savoir que l'on se trouve au cinéma à moins d'entrer explicitement l'information. Instrumenter ce téléphone est la première étape vers une technologie intelligente. Une autre étape qui fait également l'objet d'un grand nombre de travaux concerne la décision et la « conduite à tenir ». Sachant que l'utilisateur se trouve dans une salle de réunion par exemple, comment déterminer la façon de le prévenir d'un appel sans le déranger ? Cette décision requiert la connaissance de l'utilisateur, de ses habitudes, de ses préférences. Un certain nombre d'informations peut éventuellement être récupéré en instrumentant l'utilisateur (interviennent alors les vêtements intelligents).

Toutefois, il existe une étape intermédiaire entre l'acquisition des informations par les capteurs et la décision du comportement à avoir : l'interprétation des signaux acquis. Les capteurs ne fournissent que des signaux bruts et bruités à partir desquels il faut extraire les informations. Une caméra fournit une séquence d'images qui ne sont en pratique pour une machine qu'un ensemble de valeurs. Il en va de même pour tout signal acquis, que ce signal soit issu d'un son, d'un mouvement, d'un éclairage... C'est l'extraction automatique des informations utiles de ces signaux et les méthodes d'apprentissage automatique associées qui ont fait l'objet de nos premiers travaux. En effet, cette technologie intelligente qui s'adapterait aux utilisateurs ne peut exister que s'il est possible d'extraire des signaux les informations pertinentes.

Contexte : apports et défis pour l'apprentissage

L'ensemble des recherches visant au développement des outils pour une technologie sensible à l'utilisateur, à l'environnement, au contexte, est regroupé sous le terme anglais *context-aware*, qui signifie « être conscient de son contexte ». La notion de contexte est alors à définir. De manière générale, le contexte englobe tout ce qui permet d'aboutir à une décision mais qui n'est pas le stimulus direct. Pour reprendre l'exemple du téléphone portable, le stimulus direct est l'appel entrant. Dans le cas d'un appareil sensible au contexte et par conséquent instrumenté, toutes les informations issues des capteurs font partie du contexte. L'historique des actions, autrement dit la mémoire, est aussi un élément du contexte. La tâche est d'autant plus complexe que cette notion est très vaste.

Des projets de recherche tels que la prévision d'itinéraire en fonction du traffic ou encore la gestion du traffic de données en fonction à la fois de la demande, des capacités et du contenu[1], illustrent la pertinence de l'utilisation du contexte. Un autre exemple est la cuisine intelligente du MIT[2]. Celle-ci aide au réglage des appareils en fonctions des activités précédentes (par exemple, le micro-onde est automatiquement réglé sur dégivrage si la porte du congélateur vient d'être ouverte). Par ailleurs, le récent projet européen e-SENSE[3] a pour objectif de fournir des

[1] `http://amp.ece.cmu.edu/projects/`
[2] `http://web.media.mit.edu/~jackylee/kitchensense.htm`
[3] `http://www.ist-esense.org/`

réseaux de capteurs hétérogènes sans fil afin de permettre la capture du contexte.

Dans nos travaux, nous utilisons une définition plus restreinte du contexte en l'associant à la notion d'utilité [Loosli et al., 2006c]. En effet, il n'est pas nécessaire de connaitre tous les éléments du contexte, bien souvent seuls quelques éléments sont utiles. Cette fonction d'utilité est donnée par l'application visée. Par conséquent, le problème que l'on se pose n'est pas de savoir identifier n'importe quel contexte mais de reconnaître tout contexte utile.

Pour reconnaître un contexte, il faut en premier lieu le connaître, l'avoir appris. Pour savoir faire face à un nouveau contexte, il faut savoir s'adapter en se servant des éléments connus. L'apprentissage par une machine relève du domaine de l'intelligence artificielle. Plus particulièrement, nous allons étudier l'utilisation des méthodes d'apprentissage statistique pour la détection de contexte.

Dans le chapitre 1 nous cherchons à définir plus précisément ce qu'est le contexte. Au vu de la variété de contextes existants, nous restreignons notre étude à l'ensemble des contextes accessibles via des capteurs portés par une personne, tels que ceux qui sont intégrables dans les vêtements. Dans ce cadre, nous nous intéressons plus à la détection d'une forme de contexte directement lié à l'utilisateur, à ce que nous appelerons son état affectif. La démarche suivie, résolument expérimentale, consiste à étudier une instanciation du problème de la détection de contexte en espérant pouvoir généraliser les résultats obtenus. Ainsi nous décrivons les expériences que nous avons menées pour disposer de données réelles.

Nous avons mis en place une architecture de traitements allant de l'acquisition des signaux à l'étiquetage du contexte courant. Cette architecture en couche présente différentes phases de traitement et à chacune correspond une phase de « retour » qui vise à améliorer les réglages et la sensibilité de l'ensemble. La conception de cette architecture met en avant quelques problèmes difficiles en apprentissage statistique. En particulier, il devient vite évident que les données issues des capteurs sont très nombreuses. Les méthodes classiques d'apprentissage ne sont en général pas adaptées à un usage avec des grandes masses de données. En effet, on se trouve confronté à des problèmes de mémoire et de temps d'apprentissage.

En commençant par travailler sur une version simplifiée du problème, nous pouvons évaluer les points clefs à résoudre avant d'être capable de réaliser un objet qui prenne réellement en compte le contexte. Le problème simplifié consiste à travailler à partir de signaux peu bruités et pour lesquels nous connaissons le contexte associé (nous avons donc des exemples étiquetés). De plus, le temps n'est pas pris en compte. Dans cette configuration, une étude préliminaire (Loosli et al. [2003]) a montré qu'il était possible de retrouver le contexte. L'algorithme qui donne dans cette étude les meilleurs résultats en terme de reconnaissance est celui des SVM (Séparateurs à Vaste Marge, ou encore *Support Vector Machine* en anglais). Toutefois, ce résultat n'est que partiellement satisfaisant car la méthode des SVM est réputée lente et gourmande en mémoire. Par ailleurs, c'est une méthode qui a besoin de connaître l'ensemble des points avant de commencer (c'est une méthode d'apprentissage hors ligne) et qui est supervisée (tous les exemples utilisés doivent être étiquetés).

L'ensemble de ces limitations nous a conduit à approfondir l'étude de cette méthode d'apprentissage afin d'aboutir à une technique à la fois efficace en termes de reconnaissance et utilisable dans des conditions difficiles (en temps réel, avec peu de mémoire, ...).

Apprentissage autonome

L'apprentissage par SVM est une méthode de discrimination supervisée binaire. La discrimination consiste à séparer les données en fonction de leur étiquette. Dans le cas binaire, il n'y a que deux étiquettes possibles (par exemple, la tâche à effectuer peut être de dire si l'image présentée est un arbre ou un poteau). Ainsi les SVM cherchent à déterminer une frontière entre deux catégories. Cette frontière (ou fonction de décision) est tracée en fonction d'exemples connus appelés vecteurs supports. Un algorithme résolvant les SVM identifie parmi les exemples d'apprentissage quels sont les vecteurs supports et construit la frontière avec une combinaison linéaire de cette sélection. Résoudre ce problème équivaut à résoudre un programme quadratique sous contraintes de boîtes.

Dans le chapitre 2 nous abordons les diverses approches pour résoudre le problème d'optimisation posé par les SVM. Cette étude a pour objectif de répondre à un certain nombre de questions posées par l'apprentissage autonome. En particulier, est-il possible d'apprendre en temps réel ? Est-il possible d'apprendre sur un très grand nombre d'exemples ? En d'autres termes, existe-t-il des moyens d'apprendre efficacement ?

Ces questions sont complexes. Pour y apporter des éléments de réponses, l'étude porte en premier lieu sur les méthodes classiques de résolution, bien connues dans le domaine de l'optimisation, comme la méthode du point intérieur. Ensuite nous nous intéressons aux méthodes dédiées aux SVM (SMO, SimpleSVM) qui tiennent compte des spécificités du problème, la principale étant la parcimonie de la solution (en général, une faible proportion d'exemples constitue la solution).

La méthode du point intérieur se sert de tous les exemples pour construire la solution, même si certains ont une influence très proche de zéro car le fait de faire face à un problème avec une solution parcimonieuse n'est pas utilisé au cours de la résolution. En revanche, SMO ou SimpleSVM se servent de cette connaissance pour éviter des calculs inutiles. En effet, les SVM font appel à une fonction noyau qui fournit une notion de distance entre les points. Une approche naïve consiste à pré-calculer l'ensemble des valeurs correspondant à tous les couples d'exemples et à les enregistrer dans une matrice (« noyau » ou « de Gram »). Cette approche est maladroite si peu de points sont vecteurs supports car un grand nombre de valeurs sont calculées inutilement, ce qui est une perte de temps et de mémoire. D'ailleurs, pour des problèmes de grandes dimensions, cette matrice ne peut être gardée complètement en mémoire. Les méthodes modernes dédiées aux SVM évitent ce travers.

Les problèmes de grandes dimensions, en plus de l'aspect temps de calcul, posent d'importantes questions de gestion de la mémoire. Même en utilisant des techniques habiles de gestion de la matrice noyau, la mémoire devient insuffisante quand la solution contient plusieurs dizaines de milliers d'exemples (ce qui peut arriver lorsque l'ensemble d'apprentissage contient lui-même plusieurs millions d'exemples). Lorsque la mémoire est insuffisante, il faut alors recalculer les valeurs non stockées à chaque fois qu'elle sont utilisées. Dans ce cas il est utile de mettre en œuvre des méthodes qui permettent de limiter le nombre de calculs à effectuer.

Les méthodes d'apprentissage en ligne sont dans ce cadre très indiquées car les exemples ne sont considérés qu'une seule fois (contrairement aux méthodes hors ligne, qui peuvent tout à fait considérer chaque exemple un grand nombre de fois tant que la solution exacte au problème posé n'est pas atteinte). Le revers de l'approche en ligne est que la solution obtenue n'est pas la solution optimale. Nous étudions en fin de chapitre des approches non exactes de résolution des SVM. L'une est une approche pour l'apprentissage en ligne (LASVM) et l'autre une approche hors ligne mais conçue pour les grandes bases de données (CVM). Nous verrons que l'apprentissage en ligne, qui est un pré-requis indispensable à l'apprentissage en temps réel, peut être détourné pour traiter les très grandes bases de données (voir chapitre 6).

SimpleSVM

Parmi les techniques de résolution exactes des SVM, celle utilisant les contraintes actives est appelée SimpleSVM. Développée au laboratoire PSI, elle est nommée à l'origine « monqp ». En parallèle, est elle publiée par Vishwanathan et al. [2003] et est renommée en conséquence SimpleSVM. Le chapitre 3 est dédié à cette méthode de résolution et à ses déclinaisons. En effet les procédés utilisés s'appliquent à un grand nombre d'autres problèmes de programmation quadratique avec contraintes de boîte [Loosli et al., 2004]. Pour décliner les SVM, nous pouvons modifier, entre autres, les entrées et/ou les sorties.

Les sorties (ou étiquettes) souhaitées peuvent ne pas être binaires. Par exemple, il peut n'y avoir qu'une seule classe dans les exemples d'apprentissage. Dans ce cas la tâche consiste à rejeter les exemples qui ne font pas partie de la classe apprise sans chercher à savoir à quelle classe ils appartiennent. C'est le SVM à une classe ou OC-SVM (*One Class SVM*), qui sera utilisé pour la détection de rupture (voir chapitre 7). Si les étiquettes sont des réels, la tâche est alors de faire de la régression et on aboutit aux SVR (*Support Vector Regression*). Si le problème a plus de deux classes ou alors si les sorties sont plus complexes (par exemple des graphes), on parle de sorties structurées. Ces deux derniers cas restent à mettre en œuvre mais sont des extensions envisageables.

La modification des entrées intervient lorsque le problème posé présente des invariances (un chiffre a la même signification qu'il soit écrit droit ou penché) ou lorsque les entrées sont « incertaines », c'est-à-dire que la précision des appareils de mesure est connue. Le traitement des données avec invariances ou des données incertaines font souvent l'objet de plusieurs procédures séparées. Grâce à la structure du SimpleSVM, nous intégrons ces informations directement au cœur de la résolution [Loosli et al., 2005b]. Le SimpleSVM invariant est utilisé dans le chapitre 6 dans le cadre de la reconnaissance de caractères manuscrits.

Enfin, le ν-SVM, variation qui ne touche ni aux entrées ni aux sorties, reformule le problème des SVM de manière à obtenir des réglages plus intuitifs. Ce point est abordé plus en détail dans la partie suivante.

Dans ce même chapitre nous abordons également les aspects de reprise à chaud de l'algorithme : lorsqu'une solution est connue pour un problème donné et que l'on cherche une solution pour un problème proche (quelques exemples ont été ajoutés ou bien un hyper-paramètre a varié), SimpleSVM peut être initialisé avec la solution connue et donner une réponse plus rapidement. Cette démarche de reprise à chaud est particulièrement utile pour le réglage des hyper-paramètres lorsque l'on veut mettre en œuvre une validation croisée (un procédé consistant à apprendre sur des sous-ensembles de la base d'exemple pour obtenir une performance moyenne et à recommencer pour chaque valeur d'hyper-paramètre souhaitée).

Réglage automatique des hyper-paramètres grâce au chemin de régularisation du ν-SVM

En revenant au problème d'origine qui est l'utilisation de méthodes d'apprentissage dans un appareil autonome et en considérant la mise en œuvre des méthodes telles que les SVM, nous nous heurtons rapidement à un autre verrou. En effet, ces méthodes requièrent des hyper-paramètres. Les hyper-paramètres sont les valeurs que l'on fixe avant l'apprentissage pour régler la sensibilité de l'algorithme. On peut ainsi régler l'influence de chaque point, la type de noyau ou bien la sensibilité au bruit (données mal étiquetées ou bien mélangées). Ces hyper-paramètres, bien que peu nombreux dans le cas des SVM, sont un obstacle à l'autonomie d'une méthode d'apprentissage. Nous étudions ce problème dans le chapitre 4 au travers des chemins de régularisation.

L'hyper-paramètre concerné par cette méthode est celui qui règle l'influence maximale d'un point dans la solution (et donc qui règle la sensibilité au bruit). Il est noté C et influe sur la régularité de la solution. Parcourir le chemin de régularisation consiste à regarder l'ensemble des solutions pour l'ensemble des valeurs possibles de C. Cette valeur, dans les SVM, est comprise entre 0 et ∞. Pour rendre plus aisé le parcours de toutes les valeurs, il est possible de reformuler les SVM de façon à ce que l'hyper-paramètre de régularisation varie entre 0 et 1. Dans ce cas, la méthode est appelée ν-SVM et l'hyper-paramètre correspondant à C est ν.

Une fois que l'on sait parcourir l'ensemble des solutions pour tous les compromis de régularisation, on peut évaluer la qualité des solutions obtenues et sélectionner le meilleur hyper-paramètre. Nous proposons dans ce chapitre de parcourir le chemin de régularisation pour les ν-SVM et d'évaluer la qualité des solutions au fur et à mesure de l'avancement plutôt qu'à la fin. Ainsi nous pouvons arrêter l'apprentissage avant d'avoir parcouru tout le chemin (les extrêmes étant de manière générale des solutions peu recommandées).

En comparaison des approches existantes du chemin de régularisation pour les SVM, les travaux présentés ici ont l'avantage de parcourir le chemin dans le sens opposé [Loosli and Canu, 2006b]. Ainsi, alors que la méthode classique consiste à commencer avec tous les exemples vecteurs supports (ce qui implique un calcul complet du noyau), la méthode proposée commence avec le minimum possible de vecteurs supports. Si le but est de connaître toutes les solutions du chemin, cela n'apporte pas grand chose. En revanche, puisque l'on cherche à arrêter l'apprentissage en cours, il devient pertinent de ne pas calculer le noyau complet. Ainsi il est possible de traiter de plus grandes bases de données avec un chemin de régularisation.

Aspects pratiques : boîte à outils SimpleSVM

Au cours des travaux présentés ici nous avons été amenés à implémenter des méthodes regroupées dans une boîte à outils, disponible sur Internet, nommée SimpleSVM [Loosli, 2004]. Nous présentons dans le chapitre 5 cette boîte à outils avec les points clefs qui en font un outil rapide et adaptable. Nous mettons aussi en avant dans quelles conditions il est recommandé d'utiliser plutôt une méthode de résolution qu'une autre et quel est le prix des heuristiques utilisées pour accélérer la résolution des SVM.

Le passage des équations à la machine ne se fait pas sans quelques adaptations. Outre les problèmes de mémoire qui obligent à utiliser un cache (pour garder en mémoire les valeurs fréquemment utilisées et effacer les autres), on se trouve confronté aux inévitables problèmes de précision. De plus, les méthodes font souvent appel à un critère d'arrêt qui doit être respecté « à ε près ». Le choix de la valeur de ε n'est pas sans conséquence aussi bien du point de vue du temps de calcul qu'au niveau des résultats. Ces conséquences sont variables selon les méthodes de résolution et il est important de les connaître pour faire un choix sensé d'une méthode. Nous étudions ce phénomène afin d'apporter des pistes en fonction de l'ordre de grandeur des hyper-paramètres et de la taille de la base de données [Loosli and Canu, 2006a].

Invariances et très grandes bases de données

Une tâche très facile pour l'humain, voire évidente, mais complexe pour une machine est d'identifier un objet indépendamment de son éclairage, sa distance, son orientation ou sa couleur. Ce que l'on appelle l'invariance (étiquette identique malgré une transformation ou déformation de l'exemple présenté) est un problème classique de l'apprentissage. En ce qui concerne les SVM, les approches consistent la plupart du temps à effectuer un prétraitement sur les exemples connus de manière à générer des exemples déformés. De cette manière les exemples

sont plus nombreux et plus variés et les possibilités de reconnaître des variations des exemples connus sont plus élevées. Toutefois cette approche n'est pas très efficace en temps de calcul car il faut procéder à deux phases et la seconde phase doit traiter un grand nombre de points. Dans le chapitre 6 nous nous attachons à déterminer si les SVM sont à même de traiter efficacement le problème des invariances. Pour cela, il faut d'une part intégrer la génération des exemples déformés à la méthode de résolution mais aussi savoir apprendre avec un grand nombre d'exemples.

La méthode SimpleSVM est facilement adaptée aux invariances et nous décrivons comment y intégrer les transformations [Loosli et al., 2005a]. Si cette approche est plus efficace que les précédentes, elle rencontre néanmoins quelques problèmes de capacité. Du fait d'être hors ligne, SimpleSVM revient plusieurs fois sur les exemples d'apprentissage et les traitements faits pour récupérer les versions déformés doivent être recommencés à chaque fois. Une solution serait d'enregistrer toutes les transformations mais alors la mémoire est de nouveau un obstacle. Pour palier ce problème, nous nous intéressons à la méthode en ligne LASVM. En effet il s'avère que les transformations s'intègrent à LASVM aussi bien qu'à SimpleSVM [Loosli et al., 2006a,b]. Étant en ligne, chaque transformation n'est faite qu'une seule fois.

En intégrant les invariances à LASVM, nous sommes parvenus à traiter 80 millions d'exemples en 8 jours avec un seul processeur. Ce résultat est à notre connaissance le plus grand SVM reporté dans la littérature. Nous montrons donc que non seulement les invariances peuvent être traitées avec les SVM en procédant à une intégration structurelle (il faut en revanche reconnaître que d'autres méthodes comme les réseaux à convolution sont toutefois plus adaptés) mais surtout que les SVM peuvent être utilisés pour des grandes bases de données.

Détection de ruptures

Traiter les données issues de capteurs est selon nous un travail sur plusieurs couches, chacune visant à extraire de l'information pour la présenter sous une forme plus synthétique et moins bruitée à la couche suivante. De cette architecture, présentée chapitre 1 nous avons mis en œuvre la première brique consistant à segmenter un signal en morceaux homogènes. Cette segmentation est faite en détectant les changements ou les ruptures dans les signaux et elle doit être faite en ligne. L'idée employée dans le chapitre 7 est de se servir du SVM à une classe pour caractériser l'état courant du signal et déterminer l'appartenance ou non des données qui arrivent. Pour cela nous montrons comment faire des tests statistiques dans un cadre non-paramétrique.

L'étude de diverses heuristiques basées sur le test statistique est présentée sur des données synthétiques afin de déterminer une méthode efficace de détection de changement. En effet, les tests renvoient une valeur qu'il convient de comparer à un ou plusieurs seuils pour prendre une décision. Les seuils sont souvent fixés au préalable et nous comparons à ces seuils fixes des seuils mobiles qui évoluent en fonction des sorties précédentes. L'étude sur des données synthétiques ouvre la voie à une étude sur données réelles. Les données que nous utilisons sont issues de nos capteurs. Les résultats obtenus montrent qu'en pratique il est possible de segmenter des signaux à l'aide de tests non paramétriques et avec des SVM à une classe [Loosli et al., 2005c,d,e].

Contributions

Cette thèse a été débutée avec l'idée de permettre aux machines d'utiliser le contexte de l'utilisateur. Les applications sont nombreuses et variées. Du téléphone mobile à la voiture, du moteur de recherche sur Internet à la cuisine intelligente, du jeu vidéo au guide touristique, tous les domaines gagneraient à savoir reconnaître leur

contexte d'utilisation.

Très rapidement nous avons été confrontés aux limites de l'apprentissage statistique. Les techniques classiques fonctionnent surtout hors ligne (donc le temps réel n'est pas envisageable) et avec un nombre relativement restreint de données. De l'étude de l'extraction du contexte, nos travaux ont convergé vers l'étude plus approfondie des SVM. Les principales contributions concernent l'utilisation et l'implémentation des SVM pour des grandes bases de données. L'idée du *push-button* SVM, le SVM qui fonctionnerait tout seul, réglant seul ses hyper-paramètres en fonction des données et capable de gérer les masses de données arrivant en temps réel a été le fil conducteur de l'ensemble des travaux présentés ici, bien que cet ensemble puisse à première vue sembler disparate. La figure 1 montre les relations entres les différents chapitres pour aider le lecteur à relier plus facilement les éléments entre eux.

Les principales contributions concernent :

- l'identification des liens entre la détection de contexte et l'apprentissage automatique et les besoins associés en termes d'apprentissage [Loosli et al., 2006c],
- le SimpleSVM avec la mise à disposition d'une boîte à outils comprenant les SVM classiques et plusieurs dérivations ainsi qu'une étude approfondie sur les conditions d'utilisation des méthodes de résolution des SVM [Loosli and Canu, 2006a, Loosli et al., 2005b],
- le chemin de régularisation pour le ν-SVM assorti d'une méthode pour régler automatiquement l'hyper-paramètre ν [Loosli and Canu, 2006b],
- les invariances intégrées directement au sein de la résolution des SVM qui ont conduit à mettre en œuvre le plus gros SVM sur un seul processeur à ce jour [Loosli et al., 2005a, 2006a,b],
- la détection de rupture dans des signaux grâce à l'utilisation du SVM à une classe [Loosli et al., 2005c,d,e].

Le document est organisé en trois parties. La première regroupe les chapitres servant de base à l'ensemble du travail présenté (chapitres 1 et 2). La deuxième contient les apports les plus théoriques (chapitres 3 et 4) tandis que la dernière partie concerne les applications et les aspects d'implémentation (chapitres 5, 6 et 7). Les jeux de données sont présentés en annexe et les notations utilisées tout au long du document sont reportées avant la première partie (page xix).

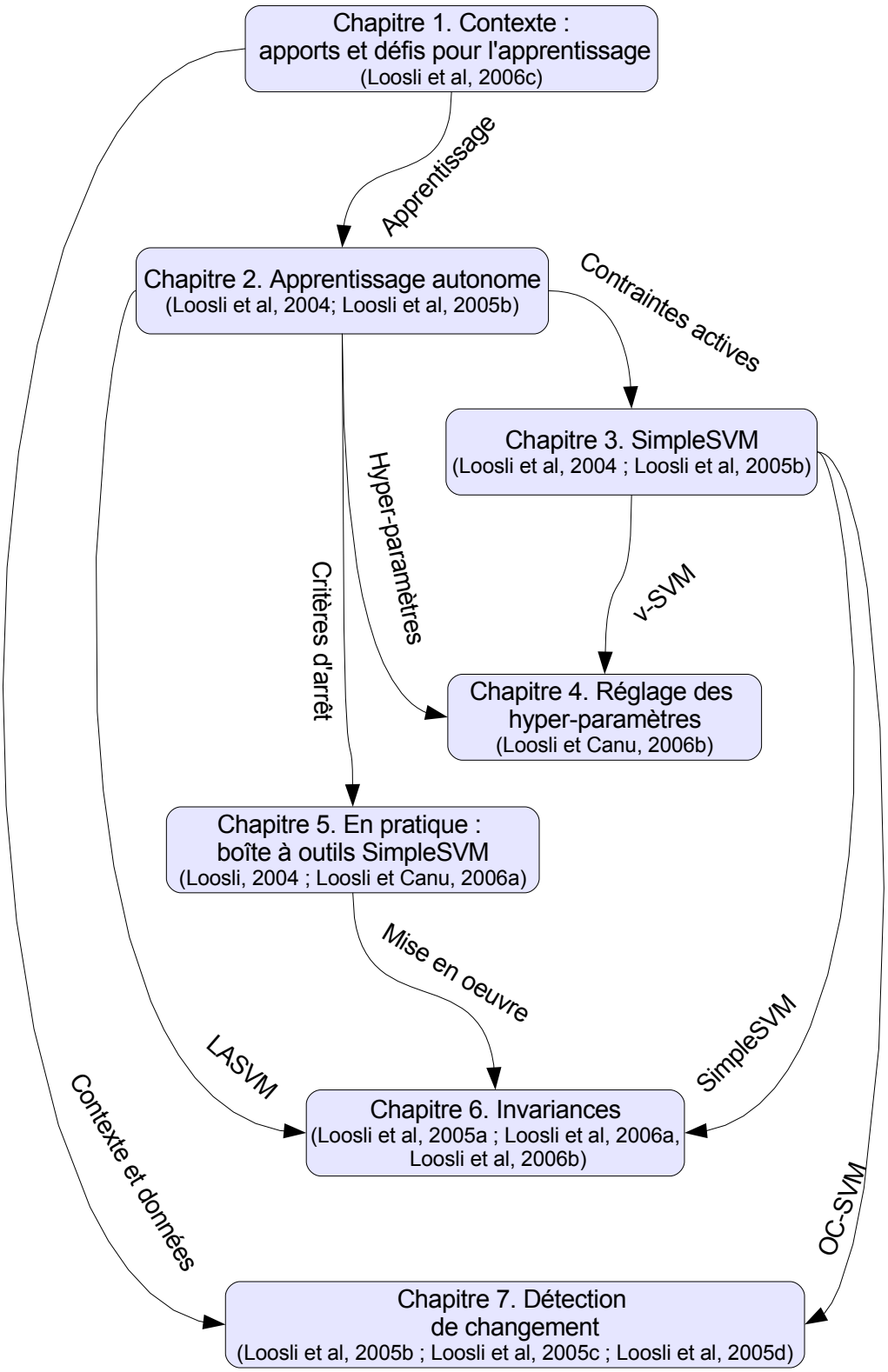


FIG. 1 : Plan de la thèse

Notations

Vecteurs De manière générale, les vecteurs sont notés en caractères gras et sont des vecteurs colonnes. Les vecteurs lignes sont indiquées par la présence du signe $'$.

$\mathbf{u}$ ..... Vecteur générique
$\mathbf{v}$ ..... Vecteur générique
$\mathbf{x}$ ..... Vecteur contenant les exemples d'apprentissage
$\mathbf{x}_i$ ..... Vecteur représentant le $i^{\text{ième}}$ exemple de la base d'apprentissage
$\mathbf{y}$ ..... Vecteur contenant les étiquettes de exemples d'apprentissage
$\mathbf{y}_i$ ..... Vecteur ou scalaire représentant la $i^{\text{ième}}$ étiquette de la base d'apprentissage
μ ..... Vecteur de multipicateurs de Lagrange issus de la résolution du problème dual
η ..... Vecteur de multipicateurs de Lagrange issus de la résolution du problème dual
ϑ ..... Vecteur de multipicateurs de Lagrange issus de la résolution du problème dual
ξ ..... Vecteur de relâchement des contraintes (SVM)
α ..... Vecteur des multiplicateurs de Lagrange (SVM)
$\tilde{\alpha}$ ..... Solution complète admissible (SVM)
$\mathbf{1}$ ..... Vecteur colonne rempli de 1
$\mathbf{0}$ ..... Vecteur colonne rempli de 0
$\mathbf{w}$ ..... Vecteur des coefficients d'un hyperplan
$\mathbf{d}$ ..... Vecteur de direction de descente

Espaces

RKHS ..... Reproducing Kernel Hilbert Space, espace de Hilbert à noyau reproduisant
$\mathscr{H}$ ..... Espace de hypothèses (RKHS)
$\mathbb{R}$ ..... Espace des réels
$\mathbb{N}$ ..... Espace des entiers
$\mathscr{X}$ ..... Espace des exemples
$\mathscr{Y}$ ..... Espace des étiquettes

Statistiques

$\mathbb{E}$ ..... Espérance
$\mathbb{P}$ ..... Loi de probabilité
$\mathscr{P}$ ..... Famille de lois de probabilité
$\mathbb{H}$ ..... Hypothèse

$\mathbb{V}\text{ar}$ ..... Variance

Fonctions

$k(.,.)$ ..... Fonction noyau
$\mathscr{C}$ ..... Fonction coût
$\mathscr{L}$ ..... Lagrangien
$\mathscr{D}$ ..... Fonction de décision

Coefficients

γ ..... Paramètre de largeur de bande du noyau
b ..... Biais ou ordonnée à l'origine d'un hyperplan
d ..... Dimension des données
n ..... Taille de la base d'apprentissage
ε ..... tolérance
C ..... Influence maximale des points dans la solution des C-SVM
ν ..... Influence maximale des points dans la solution des ν-SVM
τ ..... coefficients de déformations (invariances)
τ_b ..... coefficients de déformations (invariances)

Acronymes

SMO ..... Sequential Minimum Optimisation
SVM ..... Support Vector Machine
CVM ..... Core Vector Machine
MEB ..... Minimum Enclosing Ball

Ensembles

I_0 ..... Ensemble des indices des points dont le multiplicateur associé α vaut 0
I_C ..... Ensemble des indices des points dont le multiplicateur associé α vaut C
I_w ..... Ensemble des indices des points dont le multiplicateur associé est $0 < \alpha < C$
I_v ..... Ensemble des indices des points dont le multiplicateur associé est $0 < \alpha \leq C$
I_{cs} ..... Ensemble des indices des points du *coreset* (CVM)
I_{ncs} ..... Ensemble des indices des points extérieurs au *coreset* (CVM)

Matrices

K ..... Matrice noyau
G ..... Matrice contenant le noyau et d'autres informations (étiquettes...)

Divers

$\underset{\mathbf{x}}{arg\,max}\ S(\mathbf{x})$ . . . . . Sélection de la valeur de $\mathbf{x}$ qui donne le maximum pour l'expression S

$\underset{\mathbf{x}}{arg\,min}\ S(\mathbf{x})$ . . . . . Sélection de la valeur de $\mathbf{x}$ qui donne le minimum pour l'expression S

$\langle .,. \rangle$ . . . . . Produit scalaire

L_1 . . . . . Type de coût (absolu)

L_2 . . . . . Type de coût (quadratique)

Ω_1 . . . . . Type de pénalisation (absolue)

Ω_2 . . . . . Type de pénalisation (quadratique)

$\mathcal{O}(.)$ . . . . . Compléxité de calcul

inC . . . . . Mouvement d'un point de I_w vers I_C

$in0$ . . . . . Mouvement d'un point de I_w vers I_0

$outC$ . . . . . Mouvement d'un point de I_C vers I_w

$out0$ . . . . . Mouvement d'un point de I_0 vers I_w

Première partie

Position des problèmes abordés

1

Contexte : apports et défis pour l'apprentissage [1]

« *Les ordinateurs sont inutiles. Ils ne savent que donner des réponses.* »

Pablo Picasso

Sommaire

1.1 Apprentissage et émotions **4**
1.2 État de l'art et travaux connexes **5**
1.3 Ethique **12**
1.4 Proposition de cadre de travail **12**

LES ÉMOTIONS et les états affectifs d'une personne influent sur son comportement et ses réactions. Le stress est par exemple un facteur de changement dans le comportement, rendant les uns fébriles ou agressifs et les autres apathiques. De manière plus large, le contexte favorise ou inhibe certaines réactions et modifie les perceptions. On peut prendre pour exemple la reconnaissance visuelle d'une personne connue dans un contexte particulier. Hors de ce contexte, il devient moins facile d'être sûr d'avoir reconnu cette personne. De même, la lumière, ou l'absence de lumière, change radicalement la façon de se comporter chez la plupart des gens. Il existe ainsi une grande quantité d'exemples où le contexte est d'importance si l'on veut qu'une machine s'adapte correctement à un utilisateur.

Afin de permettre à une machine de prendre en compte le contexte de l'utilisateur, il faut lui donner un accès à son environnement, ce qui se fait par le biais de capteurs. Toutefois, avant même l'utilisation de capteurs, il faut définir quelles informations seront utiles, ce qu'est le contexte et la façon dont nous considérons une émotion ou un état affectif. Il est évident qu'il n'existe pas de réponse indiscutable à ces questions. Dans ce chapitre nous allons expliquer et définir les notions de contexte et d'état affectif que nous avons retenues et qui ont guidé nos choix, en particulier en terme de capteurs (sections 1.1 et 1.2). Ensuite une architecture ayant pour but de définir la chaîne de traitements depuis l'acquisition jusqu'à l'application est présentée section 1.4. La section 1.4.2 est dévolue à la description des deux expériences mises en place pour l'acquisition de données relatives au contexte d'un utilisateur.

[1] Ce chapitre est une version remaniée de l'article [Loosli et al., 2006c]

1.1 Apprentissage et émotions

Le détecteur de mensonges pourrait être considéré comme un bon exemple, sinon le plus ancien, de machine censée percevoir les émotions humaines. Preuve scientifique pour certains, gadget non fiable pour d'autres, la controverse qui l'entoure illustre une partie des enjeux et des problèmes liés à la conception d'un système capable de reconnaître une émotion. En effet, pour qu'un agent puisse interagir émotionnellement avec son environnement, il doit être capable de percevoir l'état affectif de son interlocuteur. Le but de nos recherches est de concevoir un système qui permette cette reconnaissance ; il pose trois questions préalables :

- les entrées - de quel type de capteurs et de mesures va-t'on disposer ?
- les sorties - comment définir l'état affectif d'une personne ?
- comment établir la relation entre les mesures et les émotions ?

Établir cette relation est difficile : il n'existe pas de modèle explicite permettant d'expliquer les émotions à partir de mesures. Cette relation est non linéaire, fortement bruitée et sujette à une grande variabilité selon les situations et les individus concernés. Enfin il est difficile de définir objectivement la notion d'émotion ; d'où notre point de départ selon lequel la perception d'émotion requiert l'utilisation de données propres à la situation et d'algorithmes d'apprentissage de type « boîte noire ». En effet, les progrès récents dans le domaine de la théorie de l'apprentissage statistique permettent d'envisager la conception de méthodes adaptatives, génériques, robustes et efficaces pour reconnaître automatiquement l'état affectif d'une personne à partir d'un ensemble de mesures.

Le but du travail présenté ici est de montrer ce que les techniques d'apprentissage statistique peuvent apporter au problème de la perception d'émotions.

Dans le domaine de l'apprentissage statistique, il est commun de distinguer le problème d'apprentissage dit « supervisé » de l'apprentissage « non supervisé ». Dans le premier cas les exemples sont étiquetés (ici des couples mesures-émotions), alors que dans le second, ils ne le sont pas (mesures seules). Il faut alors d'abord rassembler les observations en groupes homogènes (ou classes) pour pouvoir ensuite les étiqueter. Selon que l'on considère la relation entre mesures et émotions comme stable et fixée à l'avance ou sujette à d'imprévisibles dérives, l'une ou l'autre de ces approches sera utilisée. Tout dépend des « sorties » de l'algorithme d'apprentissage.

La nature de ces « sorties » fait l'objet d'une controverse entre ceux qui pensent que les émotions peuvent être détectées automatiquement [Hudlicka and McNeese, 2003, Picard, 1999] et ceux qui ne le pensent pas, considérant comme Bergson [1888, p 20] qu'il faut voir « dans l'état affectif autre chose que l'expression consciente d'un ébranlement organique ». Nous allons, dans ce travail, nous en tenir à ces « ébranlements organiques » car ils sont objectivement observables en les appelant, peut-être par abus de langage, « états affectifs ». Si le langage courant utilise indifféremment les termes « état émotionnel » et « état affectif », nous utiliserons plutôt le second en lui donnant un sens plus large et plus proche de ce que nous cherchons : affectif s'utilise *en parlant des réactions qui affectent l'être humain*. Ainsi l'état affectif désigné ici peut être d'ordre émotionnel (stressé, calme...) ou autre (marche, est assis...). De notre point de vue, la perception de ces deux types d'états est conceptuellement de même nature. De plus, puisque notre objectif est d'intégrer notre système de perception d'états affectifs dans une application pour l'améliorer, nous proposons de définir ces états affectifs en relation avec l'application considérée et plus précisément, à terme, de construire cette relation par apprentissage. Ainsi, les états affectifs que nous considérons ne sont pas subjectifs comme des sentiments personnels ou tout autre intimité réveillée. En revanche ils doivent être observables et « utiles » à l'application considérée. Il s'agit dans ce travail d'une suite d'états d'une personne pouvant être détectés (observabilité) et étiquetés dans un cadre précis.

Notre travail vise à trouver comment apprendre le lien entre ce qui affecte un utilisateur (tout en étant observable) et ce qui est utile pour une application (un programme ou une machine).

Cette définition de la nature des « états affectifs » à détecter conditionne, dans une certaine mesure, le choix des « entrées » considérées dans notre étude. En effet, les capteurs utilisés se répartissent en deux catégories selon qu'ils sont extérieurs à l'utilisateur comme une caméra ou un microphone, ou portés par lui comme une centrale d'acquisition physiologique (ECG, EMG...). Dans le premier cas, le dispositif d'acquisition est indépendant de l'utilisateur et peut être utilisé en toute occasion alors que l'utilisation de capteurs du deuxième type exige l'instrumentation de l'utilisateur. En revanche, les capteurs extérieurs nous permettent d'accéder de manière indirecte et plus subjective aux états, alors que les mesures physiologiques contiennent une part d'information objective. On sait par exemple qu'une sensation de bien-être est liée à un ralentissement du rythme cardiaque et que la colère augmente la température corporelle. C'est la raison principale qui nous a fait choisir, dans un premier temps, de ne travailler qu'avec des capteurs portés par l'utilisateur. Ce n'est toutefois pas la seule. Dans certains cas, l'utilisateur est déjà équipé (ou pourrait l'être dans un futur proche) d'un dispositif d'acquisition intégré dans ses vêtements. C'est le cas par exemple des pilotes, des secouristes, dans le domaine de la surveillance médicale ou des « cobayes » qui acceptent volontairement l'instrumentation (comme un chef d'orchestre ou un sportif souhaitant mieux se connaître). Enfin l'étude de la vision et de l'audio relève de problématiques spécifiques qui ne sont pas les nôtres ici.

Les objectifs de notre étude peuvent maintenant être précisés. Il s'agit de concevoir un système permettant d'apprendre la relation entre des signaux mesurés sur un utilisateur (physiques et physiologiques) et la suite de ses états affectifs. Cela nous amène à considérer les points suivants :

- démontrer l'utilité des techniques d'apprentissage pour ce problème et préciser le cadre de leur utilisation,
- définir quels sont les états d'une personne accessibles (observables à partir de signaux physiques et physiologiques),
- concevoir un système robuste (indépendant de l'utilisation et de l'utilisateur) permettant l'apprentissage de ces états,
- préciser quelle est l'utilité des états ainsi détectés, c'est-à-dire leur interaction avec l'application considérée.

Pour atteindre ces objectifs nous avons suivi deux stratégies différentes. D'une part nous avons recherché dans la littérature des études analogues et récupéré des données déjà traitées pour comparer différents algorithmes d'apprentissage. Ces cas ont pu être considérés comme des problèmes d'apprentissage supervisé. D'autre part, nous nous sommes dotés de notre propre système d'acquisition avec lequel nous avons pu développer et tester nos idées sur la question concernant notamment l'utilisation de l'apprentissage non supervisé. Notre approche du problème est la suivante : d'abord se donner un cadre et définir ce que nous entendons par reconnaissance d'un état affectif. Ensuite, suivant l'analyse proposée dans la huitième section de Cowie and Schroder [2005] pour aller du signal jusqu'à l'émotion, utiliser des capteurs pertinents, segmenter le signal, puis dans chaque séquence homogène, extraire des caractéristiques discriminantes permettant de l'étiqueter.

1.2 État de l'art et travaux connexes

Afin de positionner notre approche de l'apprentissage des états affectifs à partir de mesures physiques et physiologiques, nous allons rappeler l'état de l'art dans le domaine de l'apprentissage puis dans celui de la perception d'états affectifs. Nous allons voir que les enjeux croisés de ces deux domaines révèlent des difficultés fondamentales qui sont autant de blocages de la théorie de l'apprentissage et donc de ses applications [Picard et al., 2004].

1.2.1 Mise en perspective succincte de la théorie statistique de l'apprentissage

Le cadre de l'apprentissage statistique qui nous intéresse ici est celui de la reconnaissance des formes dans lequel on cherche à estimer, à partir d'un échantillon, une dépendance fonctionnelle entre des variables explicatives (observées) $\mathbf{x} \in \mathscr{X}^d$ et une variable qualitative à expliquer $\mathbf{y} \in \mathscr{Y}$.

Il est classique de distinguer les méthodes de reconnaissance des formes dites « paramétriques » des méthodes « non paramétriques »[Duda et al., 2001, Hastie et al., 2001]. Dans le cadre paramétrique, la problématique consiste à faire des hypothèses sur la nature du modèle, à en estimer les paramètres puis à vérifier ces hypothèses. Dans le cas non paramétrique, qui est aussi celui de l'apprentissage statistique tel que défini dans Vapnik [1995], on ne fait pas (du moins explicitement) référence à un modèle. On recherche des méthodes à caractère « universel » qui fonctionnent, sinon pour tout, du moins sur une large classe de problèmes. On parle aussi de modèle de type « boîte noire ».[2]

La principale qualité d'une méthode d'apprentissage est sa faculté de généralisation : son aptitude à associer en moyenne à une nouvelle observation, sinon la bonne étiquette du moins une décision raisonnable et ce, quel que soit le problème initial à résoudre. Les méthodes d'apprentissage qui nous intéressent possèdent la capacité de pouvoir approcher correctement (au sens d'un coût) n'importe quel type de fonction de décision (c'est leur caractère d'universalité). Si leur ensemble d'hypothèses doit être assez vaste pour leur permettre de tout apprendre (avec suffisamment d'exemples) c'est qu'elles peuvent tout aussi bien apprendre n'importe quoi (sur-apprentissage). Il est donc indispensable de disposer d'un mécanisme permettant d'ajuster la capacité du modèle à la complexité du problème. Les méthodes d'apprentissage statistiques se distinguent par :

- la nature de la dépendance à estimer,
- la nature des données disponibles,
- l'ensemble des hypothèses considérées,
- le critère qualifiant la qualité de l'apprentissage (un coût),
- le mécanisme de contrôle de la complexité défini comme un *a priori* sur la nature de la solution.

Afin d'illustrer certains débats actuels relatif à l'utilisation des techniques d'apprentissage, nous allons présenter une taxinomie de quelques méthodes récentes ayant démontré leur intérêt dans des applications (notamment la reconnaissance d'états affectifs).

Nous avons déjà souligné la distinction fondamentale entre apprentissage supervisé et apprentissage non supervisé. Cette distinction traite des deux premiers points de la liste. Pour schématiser, on peut dire que dans le premier cas on dispose d'un échantillon $(\mathbf{x}_i, \mathbf{y}_i), i = 1, n$, indépendant et identiquement distribué (i.i.d.) selon une loi inconnue $\mathbb{P}(\mathbf{x}, \mathbf{y})$, à partir duquel on cherche à inférer une règle. Celle-ci vise à décider à partir d'une observation $\mathbf{x}$ quelle classe lui associer. Dans le second cas, on ne dispose que d'un échantillon d'observations $\mathbf{x}_i, i = 1, n$ (sans étiquettes $\mathbf{y}_i$), toujours i.i.d. selon une loi inconnue $\mathbb{P}(\mathbf{x})$, que l'on cherche à répartir en un certain nombre de groupes à étiqueter ensuite. Lorsque l'on cherche à effectuer cet étiquetage des groupes automatiquement à partir d'autres données, on peut parler d'une certaine manière d'apprentissage « semi supervisé » bien que *stricto sensu*, l'apprentissage semi supervisé traite du cas où l'on dispose de peu de données étiquetées et d'un grand nombre d'observations non étiquetées. Il existe entre apprentissage supervisé et non supervisé un autre mode : l'apprentissage par renforcement. Il traite du cas où l'on cherche à contrôler un système (par exemple apprendre à jouer au

[2] Cette distinction n'est pas aussi simple que cela [Cornuéjols et al., 2002]. Il y a en particulier les méthodes pour lesquels il y a une forme de modèle mais ceux-ci sont mal spécifiés. Ainsi se pose alors la question de la sélection de modèle, ce que l'on retrouve dans des méthodes de type HMM, mélanges de gaussiennes ou encore dans une certaine mesure les réseaux bayésiens.

backgammon). Les observations sont des suites de données temporellement liées $\mathbf{x}_i = \{\mathbf{x}_1, ..., \mathbf{x}_{t_i}\}$ (par exemple une partie de *backgammon*) et les étiquettes une information indiquant la qualité globale de la séquence (par exemple si la partie a été gagnée ou perdue).

Théorie statistique de l'apprentissage non supervisé

Pour l'apprentissage non supervisé, l'objectif est de résumer l'information disponible. Deux approches sont possibles : soit regrouper les individus analogues (c'est une forme d'estimation de la densité de probabilité sous-jacente) soit représenter les variables par un espace de plus petite dimension. Parmi les méthodes permettant de regrouper les individus (on parle alors de création de classes, de coalescence ou de *clustering*) on trouve l'algorithme des *k*-moyennes, les nuées dynamiques, les méthodes associées aux modèles de mélanges (et à l'algorithme EM ou ses variantes) et les méthodes à « noyaux » (dont l'algorithme des *machines à vecteur support* ou SVM pour une seule classe [Scholkopf et al., 2000]). Dans ce cadre, un noyau est une fonction de deux variables mesurant d'une certaine manière une forme de leur affinité, leur corrélation ou de leur proximité. Un exemple typique est le noyau gaussien $k(\mathbf{u}, \mathbf{v}) = \exp\left(-\gamma\|\mathbf{u}-\mathbf{v}\|^2\right)$ où γ est un (hyper)paramètre à déterminer. En utilisant les noyaux, l'apprentissage peut être vu comme un problème de sélection des individus pertinents pour le problème considéré. Cette approche a donné lieu au développement de méthodes théoriquement justifiées, pratiques et efficaces [Schölkopf and Smola, 2001].

Les méthodes de représentation des variables se subdivisent elles-mêmes en deux. On peut décrire un espace par un ensemble de points topologiquement liées (sur une grille par exemple) : c'est le cas des cartes auto-organisatrices ou cartes de Kohonen. Une autre manière de faire consiste à créer de nouvelles variables sur lesquelles on projette les données disponibles : il s'agit des méthodes factorielles comme l'analyse en composantes principales ou indépendantes et leur généralisation au cas non linéaire notamment par l'utilisation de noyaux (*kernel PCA – ICA*)[Schölkopf and Smola, 2001].

Théorie statistique de l'apprentissage supervisé

En ce qui concerne l'apprentissage supervisé, une première distinction est à faire entre les méthodes estimant des probabilités pour en déduire la fonction de décision et les méthodes estimant directement la fonction de décision sans passer, du moins explicitement, par les probabilités.

Parmi les approches probabilistes, les plus utilisées se distinguent suivant la nature du modèle sous-jacent. La méthode de Bayes naïve permet d'estimer des probabilités discrètes alors que les modèles de mélange associés à l'algorithme EM sont utilisés pour l'estimation de densités et de lois mixtes. Les méthodes à noyaux ont aussi été utilisées dans le cadre bayesien sous différentes formes : processus gaussiens, *relevance vector machines* ou *Bayes point machines* [Herbrich, 2002]. Lorsque le modèle est plus complexe, la question centrale est celle de l'indépendance conditionnelle des variables. Le formalisme des modèles graphiques a été développé pour représenter ce type de relation, voir par exemple [Jordan, 2004] pour une présentation détaillée de ces modèles et des algorithmes associés pour l'estimation des probabilités. En particulier si l'on s'intéresse à la prise en compte des contraintes temporelles, les différents modèles graphiques employés sont les réseaux bayèsiens dynamiques (RBD) qui sont des graphes orientés, les modèles de Markov caché (MMC) qui en plus vérifient l'hypothèse de Markov, les modèles de Markov d'entropie maximum, et les champs aléatoires conditionnels (CAC ou CRF en anglais). Les approches du type CRF ont été introduites récemment pour résoudre le problème « étiquette-biais » lié à l'utilisation de modèles markoviens [Wallach, 2002]. Le problème typique partiellement résolu grâce à l'utilisation des modèles de Markov est celui du traitement de la parole. Soulignons aussi, pour prendre en compte le temps,

l'utilisation du filtre de Kalman là encore généralisé avec des noyaux [Ralaivola and d'Alché Buc, 2005]. Dans ce type de modèle, l'apprentissage est réalisé par l'estimation de probabilités et aussi parfois par l'évolution du modèle (ajout ou suppression d'un état). Il peut être argumenté qu'il ne s'agit pas là de méthodes d'apprentissage à proprement parler mais plutôt de méthodes permettant la modélisation d'un phénomène et donc l'introduction de connaissance *a priori* dans la solution.

Une manière de présenter les méthodes d'apprentissage supervisé dites « directes » consiste à les classer suivant la nature de leur ensemble d'hypothèses. Conceptuellement, cet ensemble d'hypothèses peut être vu comme l'ensemble de combinaisons linéaires de fonctions de référence [Hastie et al., 2001]. On distingue trois catégories selon la nature des fonctions de référence. Soit elles sont fixées indépendamment de l'échantillon en prenant une base (d'ondelettes par exemple). Soit elles ne dépendent que des observations $\mathbf{x}_i$ et pas des étiquettes. C'est le cas des méthodes à base de noyaux et notamment des séparateurs à vaste marge (SVM), des k-plus proches voisins, des arbres de décisions, des méthodes additives et celles d'agrégation comme le *boosting*. Enfin les fonctions de références peuvent dépendre à la fois les observations et des étiquettes comme pour les réseaux de neurones de type « perceptron multicouche » ou « fonction de base radiale ». Les méthodes du deuxième type conduisent à des algorithmes d'apprentissage rapides et souvent bien posés alors que les réseaux de neurones sont connus pour être associés à des algorithmes d'apprentissage au comportement chaotique. En revanche ils donnent souvent des modèles plus concis que les algorithmes de la deuxième catégorie. Une des problématique de recherche dans le domaine de l'apprentissage est donc de trouver des modèles parcimonieux facilement identifiables. C'est le cas des SVM, ce qui explique en partie leur succès actuel.

Le débat reste d'actualité entre les tenants des approches probabilistes et ceux qui prônent les méthodes directes. Si les applications ont dicté leur choix (probabiliste pour la reconnaissance de la parole et directe pour la reconnaissance de l'écriture) au niveau théorique deux conceptions s'opposent. D'un côté on défend les modèles probabilistes comme la manière convenable d'introduire les a priori sur la nature de la solution pour obtenir des modèles concis et de l'autre on met en avant le fait que l'estimation de probabilités est un problème plus difficile que celui qui nous intéresse *in fine* : l'estimation d'une fonction de décision [Vapnik, 1998].

Au-delà de cette question, il existe bien d'autres manières de distinguer les algorithmes d'apprentissage. A chaque ensemble d'hypothèses peuvent être associés différents critères qualifiant la qualité de l'apprentissage et différentes manières de minimiser ces critères. Parmi elles, une autre distinction peut être faite suivant que l'apprentissage est effectué en ligne d'une manière incrémentale ou hors ligne, une fois pour toute avant sa mise en œuvre (c'est le mode *batch*).

L'aspect dynamique lié à l'apprentissage incrémental pose de nombreux problèmes difficiles loin d'être résolus aujourd'hui. En effet, à partir du moment ou un modèle appris évolue dans le temps se pose le dilemme « stabilité plasticité » : comment garantir la stabilité du modèle tout en l'autorisant à se modifier pour suivre les évolutions imprévisibles d'un système. Un algorithme évoluant dans le temps doit être autonome et disposer d'une sorte d'auto calibration, un mécanisme de réglage de sa « capacité ». Par ailleurs un algorithme en ligne se doit d'être capable d' « oublier » des exemples pour rester réalisable en taille et en temps dans une situation réelle sans pour autant tout oublier. Il se pose alors le problème de la sélection des exemples pertinents ou qui peuvent le devenir. Enfin, pour des raisons d'efficacité, un algorithme en ligne est beaucoup plus pénalisé par le bruit et les exemples mal étiquetés que s'il est entraîné hors ligne. Cela incite à réfléchir différemment sur la façon de sélectionner les exemples pertinents en s'inspirant des méthodes d'*active learning* [Bordes et al., 2005a].

1.2.2 Apprentissage pour la perception d'états affectifs

Si l'idée d'utiliser des techniques de reconnaissance des formes pour identifier les états affectifs d'une personne à partir de signaux physiques ou physiologiques est ancienne [Cacioppo and Tassinary, 1990, Scherer, 1987] (voir Lisetti and Nasoz [2004] pour une revue détaillée), il faut attendre les travaux relativement récents de R. Picard et son équipe [Picard et al., 2001] pour voir les premiers résultats significatifs dans le domaine. Cette attente est symptomatique de réelles difficultés (« *the manifold of related theoretical and practical open problems* » [Pantic and Rothkrantz, 2003]) à commencer par celle de la définition de la variable à reconnaître : qu'entend-on par « état affectif » ?

L'utilisation d'une technique de reconnaissance des formes nécessite la connaissance *a priori* d'un certain nombre d'exemples étiquetés. Il faut se donner la liste des émotions à reconnaître et pour chaque émotion un certain nombre d'exemples et donc définir les états affectifs auxquels on s'adresse. On constate trois manières différentes de construire ces exemples étiquetés. Soit ils sont définis avant l'expérience, soit ils sont provoqués pendant l'expérience ou bien ils sont retrouvés après l'expérience. Pour définir les émotions avant l'expérience, certains on recours à des acteurs professionnels qui jouent les émotions [Banse and Scherer, 1996, Picard et al., 2001]. Si cette technique se justifie lorsqu'il s'agit de reconnaître des émotions dans la voix ou sur un visage, elle est, pour le moins, sujette à caution en ce qui concerne la physiologie.

D'autres ont préféré contrôler les expériences en provoquant les états affectifs sur le sujet. De nombreuses techniques ont été utilisées. Des modifications du comportement de l'ordinateur avec lequel un sujet interagit ont provoqué la frustration par des blocages intempestifs [Klein et al., 2002] ou par la perte d'une saisie fastidieuse [Qi and Picard, 2002]. Dans Nasoz et al. [2003] ce sont des films et des questions de mathématiques difficiles qui sont utilisés pour provoquer sept différentes émotions alors que dans Rani et al. [2005] c'est l'anxiété des sujets qui est provoquée par des tâches de difficulté croissante. Pour s'assurer de l'effet escompté Kim et al. [2004] ont eu recours à des événements multiples combinés (son, lumière, cognitif). Dans Bidel et al. [2003] le comportement de haut niveau à reconnaître est induit par la question à laquelle cet utilisateur doit répondre. D'autres ont utilisées des photos dont la vue est supposé déclencher certaines émotions [Haag et al., 2004]. Dans l'étude menée Li and Ji [2005] c'est la fatigue qui est l'état affectif provoqué par privation de sommeil (la même expérience étant reproduite au réveil et après 25 heures d'éveil). D'une manière générale, cette façon de procéder conduit à des données et des états dépendants de l'application ciblée. Ce qui est mesuré c'est la réaction à un stimulus et non pas véritablement un état affectif.

Une autre manière de construire des exemples étiquetés est de le demander au sujet lui-même avant, pendant ou après l'expérience. C'est par exemple le sujet qui choisit des chansons correspondant à quatre différents types d'émotions avant l'expérience [Kim and André, 2004]. Certains, comme Healey and Picard [2005], ont utilisé un questionnaire associé à un score établi *a posteriori* après visionnage de la vidéo pour mesurer le stress d'un conducteur alors que d'autres proposent des interfaces spécialisées pour saisir son humeur [Zimmermann et al., 2003]. Mais cette manière de faire n'est pas non plus une panacée car l'intervention du sujet contient en soit des risques de subjectivité. Si l'on veut absolument revenir à des étiquettes objectives, on peut s'en tenir aux comportements objectifs d'une personne [Laerhoven and Aidoo, 2001]. Mais on dispose là que d'une information non « émotive ».

Une démarche radicalement différente consiste à non plus s'intéresser aux états affectifs eux mêmes, mais à ce concentrer sur les effets de ce type d'outils, par exemple par une mesure objective de l'amélioration de système prenant en compte les états affectifs [Klein et al., 2002].

Nous allons retrouver dans notre étude ce triple étiquetage : les états affectifs provoqués par le contexte (les

différents événements d'un jeu), les états affectifs identifiés *a posteriori* par le sujet (après visionnage de l'expérience) et les états détectés par notre système. Reste à savoir quel est le bon étiquetage ? Nous pensons qu'à terme, ce sont les effets sur la qualité globale de l'application qui donneront la mesure de la pertinence des choix effectués.

Une fois le cadre expérimental précisé avec la définition des états affectifs à reconnaître, tout est loin d'être résolu car le choix d'une méthode d'apprentissage exige la réponse à d'autres questions, analogues à celles qui se posent dans les domaines connexes de la modélisation de l'utilisateur [Webb et al., 2001] ou de la reconnaissance d'émotions dans la parole et sur les visages [Pantic and Rothkrantz, 2003]. Une de ces questions est liée à la prise en compte du caractère dynamique du phénomène. En effet la modélisation des suites d'états affectifs est déjà complexe en soit. Une **première génération** de systèmes fait l'impasse sur cet aspect et considère la reconnaissance d'états affectifs sans prise en compte du temps. Dans ce cas, la reconnaissance d'un état affectif est vu comme un problème de classification multi-classes avec un nombre fixe et connu de classes.

Pratiquement toutes les méthodes d'apprentissage supervisé « directes » que nous avons mentionnées ont été utilisées dans différentes études. Il s'agit de l'analyse discriminante [Ark et al., 1999, Picard et al., 2001], l'utilisation de modèles de mélanges [Qi and Picard, 2002] avec l'algorithme EP (*expectation propagation*) et leur comparaison avec les séparateurs à vaste marge (SVM) qui sont aussi utilisés dans Kim et al. [2004]. On retrouve aussi les réseaux de neurones type perceptron multicouche [Haag et al., 2004, Nasoz et al., 2003] avec des comparaisons avec les k plus proches voisins [Nasoz et al., 2003]. Généralement les auteurs ne concluent pas sur la supériorité d'une méthode en terme de « taux de reconnaissance ». En revanche, d'autres arguments sont mis en avant pour préconiser l'une ou l'autre de ces méthodes et notamment la simplicité en terme de mise en œuvre. Webb et al. [2001] ont déjà souligné qu'il s'agit là d'une difficulté fondamentale liée à l'utilisation des méthodes d'apprentissage. Elles doivent être rapides, auto-configurables pour pouvoir d'adapter et peu gourmandes en ressources. C'est l'argument mis en avant par Qi and Picard [2002] pour rejeter les SVM qui sont jugées lentes, nécessitant des ressources importantes et ne permettant pas l'apprentissage en ligne (elle est qualifiée de méthode globale). Ce qui pouvait être vrai à l'époque de l'écriture de cet article ne l'est plus aujourd'hui car nous disposons maintenant d'algorithmes SVM rapides, adaptatifs, relativement peu gourmand en terme d'espace mémoire ([Bordes et al., 2005b, Loosli et al., 2004] et utilisés avec succès par exemple pour la reconnaissance des expressions du visage en temps réel [Littlewort et al., 2004].

De part sa nature contrôlée, ce genre de dispositif doit permettre de répondre à certaines questions préalables à la reconnaissance d'états affectifs :

- y a t'il des formes à apprendre ?
- quelle méthode employer ?
- comment choisir et, à partir de là, sélectionner des caractéristiques pertinentes ?

Si l'on peut répondre par l'affirmative à la première question [Christie and Friedman, 2004], les deux autres restent ouvertes.

Sous l'influence du domaine de la modélisation de l'utilisateur où l'on doit considérer des suites d'actions, la **deuxième génération** d'architecture d'apprentissage regroupe les méthodes prenant en compte le temps pour effectuer l'apprentissage dynamique des états affectifs à partir de signaux biologiques. Scheirer et al. [2002] ont été les premiers à proposer l'utilisation d'un modèle de Markov caché puis à suggérer celle de CRF. Par analogie avec le traitement de la parole en général et en particulier la reconnaissance d'émotions à partir de la parole ou de séquences vidéo, le problème est vu comme une tâche d'étiquetage de séquences. Les réseaux bayèsiens dynamiques on été préférés par Li and Ji [2005] dans le cadre d'une modélisation globale des émotions, de l'utilisateur et des ses actions (en relation avec l'application). Ce type d'approche, permettant la prise en compte du temps, nous

semble *a priori* mieux adapté à la modélisation de la succession des états affectifs. Toutefois une première critique peut être adressée aux modèles probabilistes. S'il est vrai qu'ils permettent de prendre en compte les connaissances *a priori* dont on dispose sur la nature des dépendances entre les variables considérées, ce que l'on cherche à apprendre c'est une fonction de décision et non pas une loi de probabilité (*cf* la discussion du paragraphe précédent sur les différentes méthodes d'apprentissage).

La seconde critique est liée au problème de variabilité intra et inter utilisateur. Si pour un utilisateur donné, dans une condition d'utilisation donnée ce type d'approche est très efficace, les performances se dégradent notablement, comme dans le cas de la reconnaissance de la parole, lorsque l'on modifie les conditions d'utilisation et que l'on change de sujet. En effet, ils font l'hypothèse que la distribution de probabilité du signal sous-jacent est stationnaire, ce qui n'est pas le cas lorsque l'on considère différents utilisateurs. Il y a là une autre question difficile posée par notre application au domaine de l'apprentissage. Comment apprendre sur certains individus (avec une loi de probabilité donnée) et généraliser sur d'autres (la loi de probabilité ayant changé).

Une réponse à cette question a été proposée par l'utilisation de cartes de Kohonen pour suivre l'évolution du comportement d'une personne [Laerhoven and Aidoo, 2001]. L'originalité de cette approche c'est l'utilisation d'une technique d'apprentissage non supervisée pour représenter l'information utile. Mais dans cette application, le caractère dynamique du problème n'est pas vraiment pris en compte. L'un des intérêts de cette méthode est de s'adapter, pour un utilisateur donné, aux changements de contextes (dérive de concept) et de permettre le passage d'un utilisateur à un autre. Un autre avantage lié à l'utilisation d'un modèle non supervisé, est que l'on n'est plus obligé de spécifier *a priori* tous les états que l'on souhaite détecter. Nous pouvons définir ceux que nous connaissons et laisser au système la possibilité d'en créer de nouveaux. Il se pose alors le problème de l'étiquetage de ces nouveaux états. En terme d'apprentissage, on parle de système incrémental. L'étiquetage des états est un problème lui-même complexe pour lequel nous proposerons une solution semi-supervisée dans la présentation de notre architecture. Toutefois cette proposition reste à l'heure actuelle une question à approfondir.

Pour faire face à ces problèmes, nous proposons une **troisième génération** qui n'existe pas encore et qui devrait être capable de reconnaître en temps réel une suite états affectifs indépendamment du sujet mais en adéquation et interaction avec les besoins de l'application considérée. En somme nous cherchons à obtenir un système qui aurait les caractéristiques suivantes :

- prise en compte du temps, traitement des non stationnarités du signal, de son caractère fortement bruité et du caractère multimodal des données (variabilité intra utilisateur),
- insensibilité face à la dérive de concept et adaptation au sujet (variabilité inter utilisateurs),
- efficacité, robustesse et stabilité de la mise en œuvre,
- définition évolutive des concepts à apprendre (apprentissage incrémental partiellement étiqueté),
- intégration dans l'application (association entre les états reconnus et les actions possibles et prise en compte d'un signal de qualité),

On retrouve là les questions posées par l'introduction de l'apprentissage pour la modélisation de l'utilisateur [Webb et al., 2001] en partie aussi celles évoquées pour l'apprentissage des états affectifs [Picard et al., 2004]. Plus généralement, il s'agit de questions difficiles, fondamentales et non résolues posées par l'apprentissage des facultés humaines.

1.3 Ethique

Dans un tout autre registre, il nous semble important d'évoquer les problèmes éthiques soulevés par la recherche de systèmes de reconnaissance d'états affectifs. Notre position est double : d'abord souligner l'importance de ces questions (il s'agit d'une des six principales critiques adressées aux recherches dans ce domaine [Picard, 2003]) et ensuite essayer, autant que faire se peut, d'avoir recours le plus souvent possible à des éléments objectifs. D'un point de vue juridique, la vie privée est protégée dans la loi française, européenne et onusienne. Récuperer des informations sur le corps d'une personne touche de près la sensibilité humaine. Ainsi le stockage de données privées n'est autorisé que dans des cadres relativement strictes. Les débats sur ce thèmes sont nombreux et il est difficile de déterminer des limites. Les bases de données d'empruntes digitales doivent être réservé à un usage policier.

Ces restrictions, bien que légitimes du point de vue de la vie privée, posent des problèmes quand il s'agit d'applications utilisant des données personnelles. En pratique, c'est surtout le stockage de ces données qui est contesté. Dans le cadre de la recherche, ce problème se pose pour le stockage des bases d'apprentissage. En particulier lorsqu'il s'agit de biométrie (emrpuntes digitale, iris, contour de la main, photo du visage, ...), réunir autant d'informations sur des personnes existantes pose non seulement un problème de vie privée mais aussi de sécurité [Dorizzi et al., 2004]. Pour palier ce problème, certaines équipes ont recourt aux chimères [Poh and Bengio, 2006], c'est-à-dire à des personnes artificielles.

Pour ce qui est du domaine des émotions et de leur reconnaissance, nous nous éloignons un peu des risques sur la vie privée. D'une part, nous ne cherchons pas à reconnaître des émotions précises mais une suite d'états, dont l'étiquetage nous importe moins que l'utilisation pratique que l'on peut en faire. D'autre part, nous ne stockons pas ces informations pour un usage ultérieur.

1.4 Proposition de cadre de travail

1.4.1 Architecture pour la détection de contexte humain

Nous définissons maintenant l'architecture du système de détection d'états affectifs. Les entrées de ce système sont d'une part les signaux issus des capteurs et d'autre part les étiquettes (ou informations de pertinence des réponses) données par l'utilisateur (par l'intermédiaire de l'application qui utilise les états affectifs). Le flux d'information doit donc être à double sens.

Le premier sens (figure 1.1, ascendant) doit être capable de transformer des signaux bruts en état affectif. Le deuxième sens quant à lui (descendant) doit permettre l'ajustement du flux ascendant.

Notre hypothèse de travail est qu'il est excessivement difficile de caractériser un état affectif à partir de signaux, de manière générale et applicable à tout utilisateur. *A fortiori* il est encore plus difficile de le faire pour l'ensemble des états affectifs que nous pouvons rencontrer. Par ailleurs, chaque application utilisant une connaissance de l'état affectif n'a besoin en fait que d'un nombre limité d'états pertinents. Par conséquent il n'est pas utile de chercher à étiqueter chaque état : l'apprentissage supervisé systématique n'est pas la solution.

Les flux ascendants et descendants passent par quatre modules. Chaque module est double : d'une part un traitement de l'information et d'autre part un contrôle de la méthode de traitement.

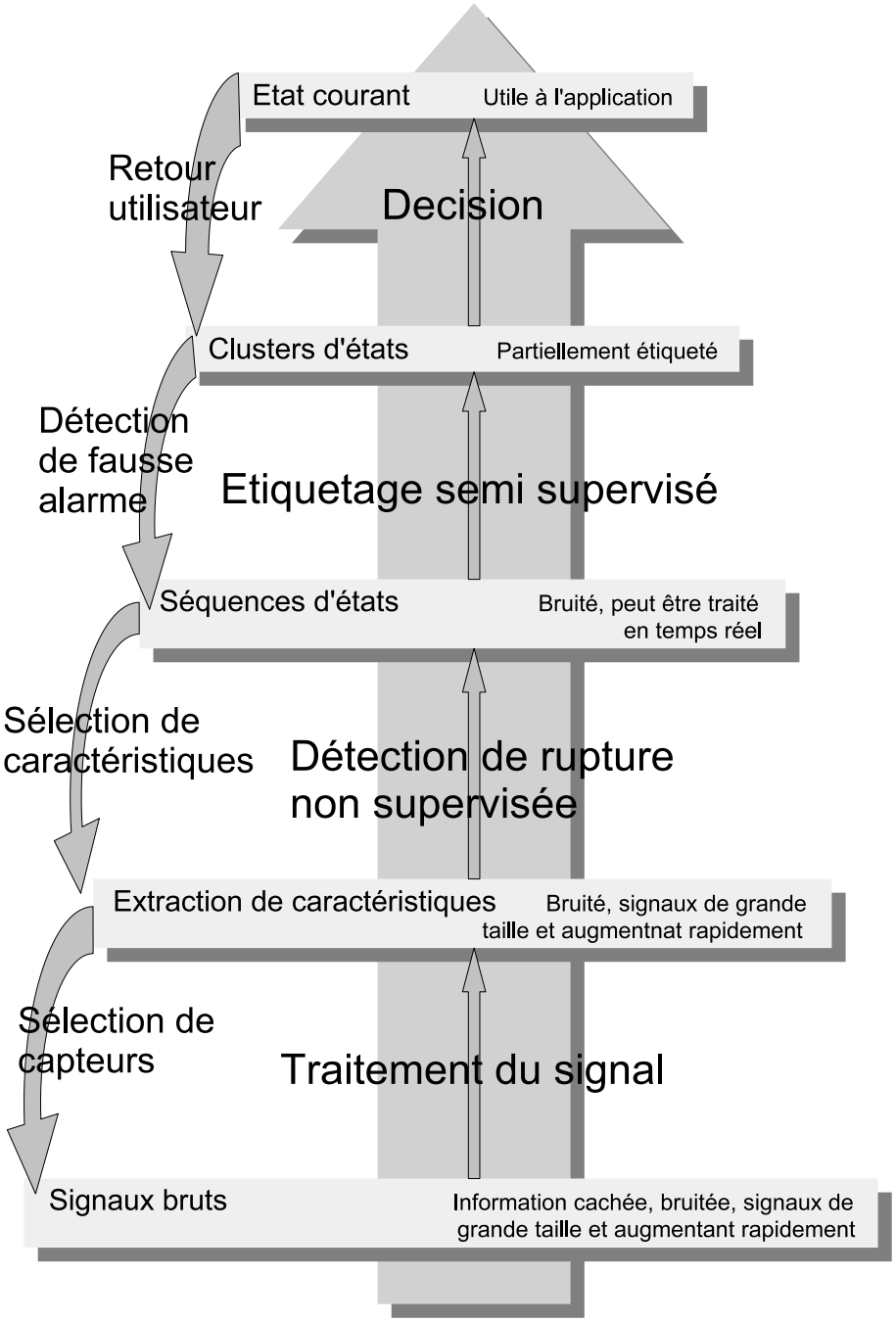


FIG. 1.1 : Architecture globale. L'état de l'utilisateur est obtenu à partir des signaux issus des capteurs qu'il porte. Les étapes intermédiaires visent à simplifier et extraire l'information reçue

Du signal vers les caractéristiques et sélection de capteurs

- Ascendant : Des signaux bruts aux caractéristiques. *La construction des caractéristiques est idéalement faite pour tout type de capteurs. Cette phase est référée comme « pré-traitements ».*
- Descendant : Sélection de capteurs. *L'élimination des caractéristiques issues d'un capteur par le module 2 peut conduire à l'élimination pure et simple du capteur - défectueux par exemple.*

Des caractéristiques vers une séquence d'états et la sélection de caractéristiques

- Ascendant : Des caractéristiques à la séquence d'états non étiquetés. *La détection de changement permet de résumer l'information des capteurs de manière non supervisée - détails dans la section suivante*
- Descendant : Sélection de caractéristiques et adaptation des paramètres de la détection de rupture. *L'information par le module 3 d'une mauvaise rupture permet d'ajuster la sensibilité de la détection ou d'identifier quelle caractéristique induit en erreur - ce qui conduit à son élimination.*

De la séquence d'états vers un réseau d'états et la détection de mauvais séquençage

- Ascendant : De la séquence d'états au réseau d'états regroupés partiellement étiquetés. *Le regroupement des segments similaires (clustering) permet d'organiser l'information et d'étiqueter les états similaires à des états précédemment étiquetés*
- Descendant : Détection de fausse alarme et de non détection. *Les fausses alarmes sont caractérisées par l'attribution des segments précédents et suivants au même regroupement. Les non-détections peuvent être remarquées par un retour de l'utilisateur qui donne une étiquette non concordante. De la même manière, le critère de regroupement peut être ajusté.*

Du réseau d'états vers l'application et l'étiquetage d'états

- Ascendant : Des états regroupés à l'application. *Dépendant de l'application, informer en permanence de l'état courant ou bien prévenir d'un état particulier lorsqu'il est détecté.*
- Descendant : Etiquetage et indication de pertinence. *Retour de l'utilisateur, dépendant de l'application.*

1.4.2 Acquisition des données réelles

Afin de pouvoir mener des études avancées sur l'apprentissage de contexte à partir de signaux biologiques, nous avons mis en place en protocole d'acquisition de données. Nous avons pour cela utilisé des capteurs biologiques et des accéléromètres *portables*. Les capteurs biologiques sont des capteurs reliés à un boîtier d'acquisition appelé Procomp (figure 1.3). Les trois accéléromètres tri-axiaux sont reliés à un autre boîtier d'acquisition, originellement conçu pour l'acquisition de signaux décrivant la démarche des sujets dans un contexte hospitalier (figure 1.2). *Le ProComp* est un système d'acquisition sur lequel nous disposons de cinq capteurs biologiques, à savoir un capteur de conductance de la peau, un capteur de respiration, de pression sanguine, un électromyogramme et un capteur de température périphérique. Dans un cadre médical ou pour rechercher une information spécifique, les capteurs doivent être placés à des endroits soigneusement choisis et la personne équipée doit veiller à ne pas faire de mouvements qui perturbent les signaux. Dans notre approche nous cherchons à reproduire des données réalistes pour une utilisation dans la vie quotidienne. Par conséquent l'utilisateur ne doit pas « prendre garde » aux capteurs lors des prises de données. Ces systèmes d'acquisition sont décrits plus en détails en annexe (7.3.5).

Utilisateur monitoré dans un contexte anodin

L'utilisateur est équipé des cinq capteurs biologiques ainsi que des trois accéléromètres tri-axiaux. Il peut ensuite aller et venir à sa convenance (figures 1.2 et 1.3).

FIG. 1.2 : Recueil de données sur une personne se déplaçant dans les couloirs

FIG. 1.3 : Système d'acquisition Procomp

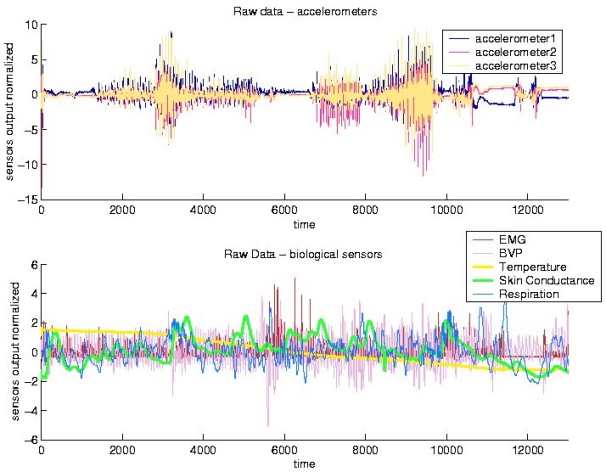


FIG. 1.4 : Données acquises

Utilisateur monitoré dans le cadre d'un jeu vidéo

Pour cette expérience nous n'utilisons que les capteurs biologiques, car l'utilisateur reste assis (figure 1.5). On enregistre les signaux pendant qu'il joue à un jeu vidéo impliquant des réflexes et créant joie ou frustration. Le jeu (Xblast) se joue en réseau et consiste à poser des bombes pour tuer tous ses adversaires avant la fin du temps imparti. Une caractéristique intéressante du jeu est son organisation par niveau. Au cours de la partie, le joueur ne joue pas tout le temps s'il ne gagne pas tous les niveaux. Une des choses que l'on va donc chercher à retrouver dans les signaux est l'état « en jeu/hors jeu », ce qui se traduit dans les ruptures par « commence à jouer/arrête de jouer ». Les autres ruptures qui sont relevées et que l'on essaiera de retrouver sont les événements tels que « tuer/blesser un adversaire » ou « être tué/blessé ».

FIG. 1.5 : Recueil de données sur un joueur de jeux vidéos

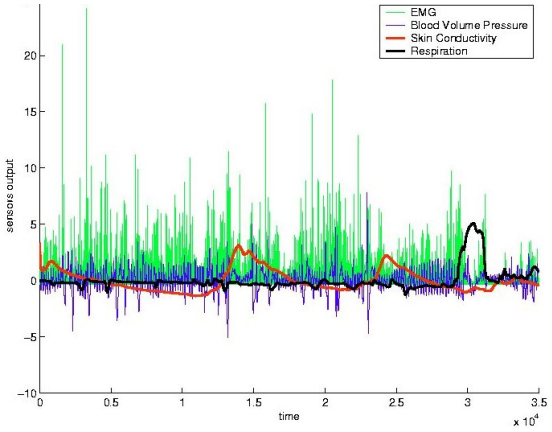


FIG. 1.6 : Données acquises

Conclusion

Dans ce nous avons défini notre idée du contexte. Au vu de la variété des contextes existants, nous avons restreint notre définition au contexte affectif et utile. Pour identifier le contexte d'une personne, nous avons choisi une approche expérimentale basée sur deux expériences. L'une nous fournit des données multimodales issues de capteurs physiologiques et d'accéleromètres, l'autre uniquement des données physiologiques. L'architecture destinée à traiter ces données n'est pas entièrement réalisée au cours de la thèse présentée ici. Parmi les modules présentés, nous avons plus particulièrement travaillé sur la détection de changement dans les signaux (chapitre 7). Toutefois, isoler les tâches met à jour des blocages induits par l'état d'avancement des méthodes d'apprentissage automatique.

Nous avons besoin de méthodes autonomes et endurante et celles-ci ne sont pas encore disponibles. Pour cette raison nous présentons dans les chapitres suivants des travaux moins spécifiques à la détection de contexte mais utiles à long terme.

Nous nous sommes posé la question de l'apprentissage automatique pour une utilisation grand publique, dans la durée et sans supervision. Les contraintes de mémoire, de temps de calcul, de réglage sont autant de problèmes. Le chapitre suivant présente différentes méthode d'optimisation pour les SVM puisque c'est l'algorithme d'apprentissage que nous avons retenu pour l'ensemble de nos travauxLoosli et al. [2003].

2

Apprentissage automatique et optimisation quadratique [1]

« Why waste time learning,
when ignorance is instantaneous ? »

Hobbes

Sommaire

2.1 L'apprentissage, aspects théoriques 18
2.2 L'optimisation quadratique sous contraintes pour l'apprentissage 22
2.3 Méthodes alternatives non optimales 26
2.4 Résumé des méthodes de résolution 29

APPRENDRE, POUR UN ENFANT, est un processus complexe qui vise à acquérir ou à développer certaines facultés à partir d'expériences diverses. Pour une machine, cela se réduit à utiliser des exemples ou des observations pour améliorer ses performances. Ainsi, l'apprentissage automatique peut se concevoir comme l'optimisation, dans un ensemble suffisamment grand, d'un critère défini à partir d'exemples ; plus il y aura d'exemples, meilleur devra être l'apprentissage. Cette manière de voir pose, du point de vue de l'optimisation, des problèmes spécifiques : comment définir le critère à optimiser, comment gérer la masse des données, comment trouver un algorithme efficace dans ce cadre ?

Pour traiter ces problèmes, les méthodes d'apprentissage de type réseaux de neurones proposent d'utiliser des critères non convexes associés à des méthodes de descente de gradient. Ce procédé entraîne de nombreuses difficultés pratiques liées à la non convexité du critère à optimiser. L'une des clés du succès des méthodes à noyaux, acquis une dizaine d'années après l'introduction des réseaux de neurones, est leur capacité à formuler le problème d'apprentissage comme un problème de programmation quadratique (convexe) de grande taille et très particulier. En effet, avec les méthodes à noyaux, la solution recherchée est souvent « parcimonieuse » : un très grand nombre de ses composants sont nuls. Profitant aujourd'hui de cette spécificité, les algorithmes d'apprentissage peuvent résoudre en un temps raisonnable des problèmes de programmation quadratique de très grande taille : 1

[1] Ce chapitre est une version remaniée et augmentée du chapitre de livre « Optimisation en traitement du signal et de l'image » à paraître aux éditions Hermès. Le chapitre s'intitule *Programmation quadratique et apprentissage* et est co-signé Gaëlle Loosli et Stéphane Canu.

journée de calcul en moyenne avec une machine mono-processeur pour 8 millions d'inconnues. Sur ces 8 millions d'inconnues, seules de l'ordre de 8 000 à 20 000 sont non nulles en fonction de la complexité du problème.

Ce chapitre traite de la programmation quadratique pour l'apprentissage. Si dans le cas général de la programmation quadratique les méthodes de type « point intérieur » semblent être les plus efficaces, dans le cas particulier de la programmation quadratique pour l'apprentissage ce sont les méthodes des « contraintes actives » qui vont se révéler les plus performantes. Cela est dû au fait que les méthodes de contraintes actives utilisent avantageusement la géométrie spécifique du problème d'apprentissage et la nature parcimonieuse de la solution recherchée.

Les algorithmes issus des contraintes actives donnant la solution exacte du problème posé ont besoin de connaître à l'avance l'ensemble des points d'apprentissage. Ce fonctionnement est appelé « hors ligne », ou en mode « batch ». Toutefois tous les problèmes d'apprentissage n'entrent pas dans ce cadre. Il peut arriver que l'on ait connaissance des exemples pendant l'apprentissage. On parle alors d'apprentissage « en ligne ». L'apprentissage en ligne devra être aussi employé quand l'ensemble d'apprentissage, même connu entièrement à l'avance, est trop grand pour qu'il soit envisageable de prendre en compte simultanément tous les exemples disponibles pour assurer l'optimalité. Le cadre en ligne a engendré le développement de méthodes d'optimisation itératives stochastiques particulièrement adaptées aux problèmes de grande taille à solution parcimonieuse. Il est communément admis qu'aucune méthode à convergence polynomiale (donc en particulier les programmes quadratiques) ne peut être utilisée pour résoudre des problèmes de très grandes dimensions [Nemirovski, 2005]. Néanmoins, si cela peut être vrai lors de la recherche de solutions exactes, il est possible de faire évoluer les méthodes vers des algorithmes qui donneront une solution certes non optimale (ce qui est négligeable compte tenu des incertitudes liées aux données) mais bien plus rapidement. Nous présentons à cet effet une approche stochastique en ligne, qui permet de résoudre à la fois les problèmes en ligne et de très grandes dimensions et ainsi de dépasser les limitations des méthodes hors ligne.

Ce chapitre est organisé de la manière suivante. Il commence par la description du cadre général de l'apprentissage et des outils permettant de le voir comme de la programmation quadratique convexe. Nous verrons ensuite comment les méthodes de contraintes actives utilisent avantageusement la géométrie et la parcimonie de cette formulation, puis nous décrirons une méthode d'optimisation itérative stochastique très efficace baptisée LASVM ainsi qu'une méthode de résolution approchée, CVM.

2.1 L'apprentissage, aspects théoriques

L'apprentissage statistique de données utilise des vecteurs forme $\mathbf{x}$ de dimension d. Dans le cas de l'apprentissage supervisé, chaque individu $\mathbf{x}$ est accompagné de son étiquette $\mathbf{y}$. Cette étiquette permet de caractériser l'individu (son groupe d'appartenance, sa valeur, sa structure...). Le but de l'apprentissage est de permettre à une machine de retrouver l'étiquette pour un élément donné.

2.1.1 Cadre général

On dispose d'une famille $\mathscr{P}$, l'ensemble des lois sur $(\mathbb{R}^d \times \{-1,1\})$. Dans cet ensemble, un élément $\mathbb{P}(\mathbf{x},\mathbf{y})$ est la loi de probabilité inconnue selon laquelle les points de la base d'apprentissage sont générés. Dans ce cadre, le résultat de l'apprentissage est une fonction f (définie plus précisément dans la partie 2.1.2) qui, pour n'importe quel $\mathbb{P} \in \mathscr{P}$, est capable de bien classer en moyenne les points issus de $\mathbb{P}$ et ce avec grande probabilité selon l'échantillon.

DÉFINITION - LA BASE D'APPRENTISSAGE : $\left\{\mathscr{X}_n^d, \mathscr{Y}_n\right\}$ regroupe l'ensemble des exemples connus générés par la loi sous-jacente inconnue $\mathbb{P}(\mathbf{x},\mathbf{y}) \in \mathscr{P}$. La taille de cette base est notée n et chaque exemple (ou individu) est noté $\mathbf{x}_i$, avec $i \in [1,n]$. Les exemples appartiennent à un espace de dimension d (le plus souvent $\mathbb{R}^d$). Les étiquettes associées à chaque exemple sont la plupart du temps représentées par des scalaires. Les cas les plus fréquents sont $y_i \in \mathbb{R}$ pour la régression et $y_i \in \{-1,1\}$ pour la discrimination.

DÉFINITION - LA BASE DE TEST : $\left\{\mathscr{X}_t^d, \mathscr{Y}_t\right\}$ représente un ensemble quelconque d'exemples générés par la même loi $\mathbb{P}(\mathbf{x},\mathbf{y}) \in \mathscr{P}$.

Un algorithme est efficace s'il est capable de donner des résultats similaires (en termes d'erreur) à la fois sur la base d'apprentissage et sur la base de test. C'est ce que l'on appelle la capacité à généraliser.

DÉFINITION - LE RISQUE EMPIRIQUE : donne la mesure de l'efficacité d'un algorithme sur la base d'apprentissage en fonction d'une fonction coût $\mathscr{C}$ donnée.

$$R_{emp}\big[f\big] = \frac{1}{n}\sum_{i=1}^{n} \mathscr{C}(f,\mathbf{x}_i,\mathbf{y}_i) \quad \text{avec par exemple : } \mathscr{C}(f,\mathbf{x}_i,\mathbf{y}_i) = \frac{1}{2}\big|f(\mathbf{x}_i) - \mathbf{y}_i\big|$$

Le risque empirique est l'une des quantités que l'on cherche à minimiser lors de l'apprentissage. A priori, rien ne garantit qu'une fonction f qui donne un risque empirique faible aura de bons résultats sur des données jamais vues : il faut contrôler la capacité de généralisation et par conséquent tenir également compte du *risque* lors de l'apprentissage.

DÉFINITION - LE RISQUE : est la notion similaire au risque empirique mais sur les données de test. Comme ces données sont inconnues, nous avons besoin de $\mathbb{P}(x,y)$ pour exprimer le risque :

$$R\big[f\big] = \mathbb{E}\big(\mathscr{C}(f,X,Y)\big) = \int \mathscr{C}(f,\mathbf{x},\mathbf{y}) d\mathbb{P}(\mathbf{x},\mathbf{y})$$

Pour ce faire, on minimise en même temps que le risque empirique une autre quantité permettant de contrôler cette erreur en généralisation. La théorie statistique de l'apprentissage (voir Schölkopf and Smola [2002], Vapnik [1995] pour plus de détails) utilise la notion de *capacité* pour exprimer cette quantité.

DÉFINITION - LA CAPACITÉ : d'un algorithme est définie comme étant le cardinal de l'espace des hypothèses. Dans le cas où cet espace est de dimension infinie, la capacité est définie par la VC-dimension.

La VC-dimension donne une notion de capacité d'un ensemble de fonctions. Elle est définie en indiquant combien de points peuvent être séparés de toutes les manières possibles par les fonctions de cet ensemble. A partir de cette VC-dimension, la théorie de Vapnik-Chervonenkis fourni des bornes sur le risque. On peut ainsi borner le risque en fonction du risque empirique, de la capacité des fonctions utilisées (leur VC-dimension h) et du nombre d'exemples de la base d'apprentissage, avec une probabilité de $1-\eta$ pour $\mathscr{C} = \frac{1}{2}|f(x_i) - y_i|$, [Burges, 1998, équation 3, page 3] :

$$R\big[f\big] \leq R_{emp}\big[f\big] + \sqrt{\frac{h(log(2n/h)+1) - log(\eta/4)}{n}}$$

L'idée est que les fonctions obtenues par apprentissage doivent être suffisamment *flexibles* pour pouvoir apprendre les données mais aussi assez *régulières* pour être capables de généraliser. Apprendre est alors vu comme un problème d'optimisation bi-critères : trouver la fonction de plus faible capacité minimisant le risque empirique.

2.1.2 Cadre fonctionnel

Un algorithme d'apprentissage permet de choisir, dans un ensemble d'hypothèses $\mathscr{H}$, la meilleure solution compte tenu de la base d'apprentissage et des deux critères à optimiser. Les méthodes à base de noyaux construisent cet ensemble d'hypothèses $\mathscr{H}$ à l'aide d'un noyau. Intuitivement, pour classer des points, il est utile de disposer d'une notion de proximité ou de ressemblance entre les objets que l'on cherche à séparer. Si l'on sait donner une distance entre deux objets, quelque soit leur forme, on peut se contenter de cette information pour réaliser la discrimination. En partant de cette idée, nous définissons les *noyaux* :

DÉFINITION - UN NOYAU : est une fonction symétrique de deux variables qui retourne un scalaire exprimant une notion de distance entre les variables. Plus formellement, soient $\mathbf{u},\mathbf{v} \in \mathbb{R}^d$, on défini $k(.,.)$ comme étant :

$$\begin{array}{rccc} k(.,.) & : & \mathbb{R}^d \times \mathbb{R}^d & \rightarrow & \mathbb{R} \\ & & \mathbf{u},\mathbf{v} & \mapsto & k(\mathbf{u},\mathbf{v}) \end{array}$$

A partir du noyau (lorsqu'il est défini positif[2]), on peut construire l'espace des hypothèses en passant par l'espace pré-hilbertien $\mathscr{H}_0$.

DÉFINITION - ESPACE PRÉ-HILBERTIEN. : Soit $k(,)$ un noyau défini positif. On définit $\mathscr{H}_0$ l'espace vectoriel engendré par les combinaisons linéaires finies du noyau :

$$\mathscr{H}_0 = \left\{ f : \mathbb{R}^d \mapsto \mathbb{R} \;\middle|\; \ell < \infty, \{\alpha_i\}_1^\ell \in \mathbb{R}, \{\mathbf{u}_i\}_1^\ell \in \mathbb{R}^d, f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i k(\mathbf{u}_i,\mathbf{x}) \right\}$$

On donne à cet espace le produit scalaire défini par la forme bilinéaire telle que pour tout couple $f,g \in \mathscr{H}_0, f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i k(\mathbf{u}_i,\mathbf{x})$ et $g(\mathbf{x}) = \sum_{j=1}^{m} \beta_j k(\mathbf{v}_j,\mathbf{x})$,

$$\langle f,g \rangle_{\mathscr{H}_0} = \sum_{i=1}^{\ell}\sum_{j=1}^{m} \alpha_i \beta_j k(\mathbf{u}_i,\mathbf{v}_j)$$

La positivité du noyau garantit la positivité du produit scalaire ainsi défini. L'espace engendré muni de ce produit scalaire est un pré-hilbertien. La complétion de l'espace pré-hilbertien $\mathscr{H}_0$ au sens de la norme induite par le produit scalaire, est un espace de Hilbert noté $\mathscr{H} = \overline{\mathscr{H}_0}$.

PROPRIÉTÉ : La norme induite dans cet espace est $\|f\|^2_{\mathscr{H}} = \langle f,f \rangle_{\mathscr{H}}$. Par ailleurs, on remarque que :

$$\langle f(.),k(.,\mathbf{x}) \rangle_{\mathscr{H}} = \left\langle \sum_{i=1}^{\ell} \alpha_i k(\mathbf{u}_i,.),k(.,\mathbf{x}) \right\rangle_{\mathscr{H}} = f(\mathbf{x})$$

C'est la propriété de reproduction. L'espace $\mathscr{H}$ est donc un espace de Hilbert à noyaux reproduisant, aussi appelé *RKHS* (Reproducing Kernel Hilbert Space).

Grâce à l'utilisation judicieuse de la propriété de reproduction, le problème de recherche d'une solution générale f dans l'espace de fonctions $\mathscr{H}$ se ramène à la recherche d'un vecteur $\alpha \in \mathbb{R}^n$ où n désigne le nombre d'exemples disponibles pour l'apprentissage. Le lien entre la fonction f et le vecteur α est donné par un théorème dit de représentation (théorème 4.2 page 90 [Schölkopf and Smola, 2002]) :

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i,\mathbf{x})$$

[2]Nous ne considérerons que le cas des noyaux définis positifs *i.e.* tels que $\forall \ell < \infty, \forall \alpha_i \in \mathbb{R}, \forall \mathbf{x}_i \in \mathbb{R}^d, i = 1...\ell$; $\sum_i \sum_j \alpha_i \alpha_j k(\mathbf{x}_i,\mathbf{x}_j) \geq 0$.

2.1.3 Coût et Régularisation

Maintenant que nous avons défini l'ensemble des hypothèses, nous allons pouvoir préciser les deux critères minimisés lors de l'apprentissage : le risque empirique et la capacité. On parle alors de minimisation de coût régularisé.

La *régularisation* est une forme du contrôle de la capacité. L'idée est d'utiliser un terme de pénalisation. Ce terme croît quand la fonction de décision augmente en complexité. Pour que la solution exacte du problème soit parcimonieuse il est nécessaire qu'au moins l'un de ces deux critères soit singulier (par opposition aux coûts réguliers) [Nikolova, 2000][3].

DÉFINITION - COÛT RÉGULIER. : Un coût est dit régulier si il est infiniement dérivable.

C'est le cas par exemple du coût $\mathscr{C}_2 = (f(\mathbf{x}_i) - \mathbf{y}_i)^2$ utilisé pour la régression, du coût associé à la régression logistique $\mathscr{C}_\ell = \mathbf{y}_i \log f(\mathbf{x}_i) + (1-\mathbf{y}_i)\log(1-f(\mathbf{x}_i))$ et des termes de pénalisation quadratiques du type $\Omega_2 = \|f\|^2_{\mathscr{H}}$.

Les coûts singuliers sont ceux qui conduisent à la parcimonie. Il sont souvent construits à partir de valeurs absolues et sont singuliers à l'origine [Nikolova, 2000]. C'est le cas par exemple des coûts $\mathscr{C}_1 = |f(\mathbf{x}_i) - \mathbf{y}_i|$ et $\mathscr{C}_1(\varepsilon) = \max(0, |f(\mathbf{x}_i) - \mathbf{y}_i| - \varepsilon)$ utilisés pour la régression, du coût charnière $\mathscr{C}_h = \max(0, \mathbf{y}_i(f(\mathbf{x}_i) + b - 1))$ utilisé en discrimination et des termes de pénalisation du type $\Omega_1 = \sum_{i=1}^n |\alpha_i|$ lorsque la solution f que l'on recherche vérifie $f(\mathbf{x}) = \sum_{i=1}^n \alpha_i k((,\mathbf{x})_i, \mathbf{x})$.

Le choix des fonctions de coût et de régularisation conduit à différents algorithmes (voir le tableau 2.1 pour quelques exemples). Les algorithmes minimisant un coût régulier sont de type Gaussien et sont très populaires pour leurs propriétés de dérivation et de facilité de calculs. Les algorithmes minimisant un terme singulier sont quant à eux très recherchés pour leur capacité à donner des solutions parcimonieuses. En revanche, l'utilisation de termes de type singulier impliquent des résolutions réputées lentes dans le pire des cas (on peut en particulier se référer au simplexe). Portnoy and Koenker [1997] montrent comment une méthode singulière peut être concurrentielle aux méthodes régulières d'un point de vue algorithmique. Nous allons montrer ici comment pratiquement faire mieux en tirant parti de la parcimonie.

Coût	Ω	Discrimination	Régression
sing.	sing.	LP SVM [Mangasarian, 1998]	LP SVR : $\mathscr{C}_1(\varepsilon)$ et Ω_1
rég.	sing.	Regression Logistique L_1	LARS : $\mathscr{C}_2$ et Ω_1 Efron et al. [2004]
sing.	rég	SVM : $\mathscr{C}_h$ et Ω_2	SVR : $\mathscr{C}_1(\varepsilon)$ et Ω_2
rég.	rég.	K-régression logistique : $\mathscr{C}_\ell$ et Ω_2 ; Lagrangian SVM : $\mathscr{C}_h^2$ et Ω_2 [Mangasarian and Musicant, 2001]	Splines : $\mathscr{C}_2$ et Ω_2

TAB. 2.1 : Récapitulatif de divers algorithmes d'apprentissage selon la nature de leurs fonctions objectifs.

2.1.4 Sur les objectifs d'un apprentissage réaliste

L'apprentissage se doit d'être un outil pour le traitement de données issues du monde réel. En traitement du signal en particulier, les données sont nombreuses, fortement bruitées, variables. Les défis de l'apprentissage passent donc par le développement de méthodes non seulement performantes dans leur tâches première (classification, ré-

[3] voir aussi l'article *"When does sparsity occur ?* sur le blog d'O. Bousquet `http://ml.typepad.com/machine_learning_thoughts/`

gression, ...) mais aussi (et surtout) en temps et en espace mémoire. Pour cette raison, les algorithmes en ligne et les algorithmes qui ont une complexité peu élevée (moins de $\mathcal{O}(n^2)$) sont à privilégier pour le développement d'applications réalistes.

2.2 L'optimisation quadratique sous contraintes pour l'apprentissage

Il existe de nombreuses méthodes à noyaux qui aboutissent à la résolution d'un problème quadratique sous contraintes. On peut citer dans cette catégorie le SVM, le SVM à une classe, le SVR (la regression par machines à vecteurs support), etc. Nous allons ici montrer la dérivation du SVM binaire de sa forme primale à sa forme duale, cette dernière étant un problème quadratique sous contraintes de boîtes.

2.2.1 SVM : un problème quadratique sous contraintes

Pour un problème de discrimination à deux classes, on cherche une frontière de décision entre les exemples de chaque classe. On choisi cette frontière dans un *RKHS* appelé $\mathscr{H}$. La fonction de décision est $\mathscr{D}(\mathbf{x}) = sign(f(\mathbf{x}) + b)$. La formulation primale du SVM binaire est la suivante :

$$\left\{ \begin{array}{ll} \min\limits_{f\in\mathscr{H},\xi,b} & \frac{1}{2}\|f\|^2_{\mathscr{H}} + C\sum\limits_{i=1}^{n}\xi_i \\ & y_i(f(\mathbf{x}_i)+b) \geq 1-\xi_i \quad i\in[1,...,n] \\ & \xi_i \geq 0 \qquad\qquad i\in[1,...,n] \end{array} \right. \tag{2.1}$$

Ce système permet de trouver la fonction de $\mathscr{H}$ qui a la plus petite norme L_2 (régularisation) et qui classe correctement les points d'apprentissage. La contrainte de bon classement (le coût, ici L_1) est assortie d'une marge de 1 qui oblige la frontière à se situer le plus loin possible des points d'apprentissage. On permet par ailleurs de violer la contrainte de bon classement (c'est-à-dire de relâcher les contraintes) grâce aux variables ξ_i. C représente l'influence maximale d'un point dans la solution. Résoudre un tel système passe par l'utilisation du Lagrangien :

$$\mathscr{L}(f,\xi,b) \quad = \quad \frac{1}{2}\|f\|^2_{\mathscr{H}} + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(f(\mathbf{x}_i)+b)-1+\xi_i) - \sum_{i=1}^{n}\beta_i\xi_i \tag{2.2}$$

avec pour tout $i\in[1,...,n]$, $\xi_i \geq 0, \alpha_i \geq 0$ et $\beta_i \geq 0$. Minimiser (2.1) revient alors à annuler les dérivées de $\mathscr{L}$ par rapport à chaque variable :

$$\left\{ \begin{array}{l} \Delta_f(\mathscr{L}) = 0 \\ \Delta_\xi(\mathscr{L}) = 0 \\ \Delta_b(\mathscr{L}) = 0 \end{array} \right. \iff \left\{ \begin{array}{l} f(.) = \sum\limits_{i=1}^{n}\alpha_i y_i k(\mathbf{x}_i,.) \\ C-\beta_i-\alpha_i = 0 \\ \sum\limits_{i=1}^{n} y_i\alpha_i = 0 \end{array} \right. \tag{2.3}$$

En replaçant ces relations dans (2.2), on obtient le problème dual, que l'on écrit sous forme vectorielle pour plus de clarté :

$$\left\{ \begin{array}{ll} \max\limits_{\alpha\in\mathbb{R}^n} & -\frac{1}{2}\alpha^\top G\alpha + \mathbf{1}^\top\alpha \\ & \mathbf{y}^\top\alpha = 0 \\ & 0\leq\alpha_i\leq C \qquad i\in[1,...,n] \end{array} \right. \tag{2.4}$$

G désigne la matrice noyau pondérée par les étiquettes des points, telle que $G_{ij} = y_i k(\mathbf{x}_i,\mathbf{x}_j) y_j$. Ce système dual est un programme quadratique convexe qui présente deux particularités : il y a n inconnues avec $2n+1$ contraintes et G est une matrice semi-définie positive.

2.2.2 La propriété utilisée : la parcimonie

Notons tout d'abord que les formes à optimiser sont convexes et par là même n'admettent qu'une seule solution optimale. Notons par ailleurs que les contraintes sont linéaires et en nombre fini : la solution optimale α vérifie pour chacune de ses composantes α_i les conditions de Karuch Kuhn Tucker (KKT), c'est-à-dire :

$$\alpha_i = 0 \quad \rightarrow \quad \big(f(\mathbf{x}_i)+b\big)y_i > 1 \tag{2.5}$$

$$0 < \alpha_i < C \quad \rightarrow \quad \big(f(\mathbf{x}_i)+b\big)y_i = 1 \tag{2.6}$$

$$\alpha_i = C \quad \rightarrow \quad \big(f(\mathbf{x}_i)+b\big)y_i < 1 \tag{2.7}$$

Le principe général des méthodes de décomposition et/ou de contraintes actives repose sur l'observation que seuls les points non contraints dans la solution nécessitent le calcul de leur coefficient : c'est la parcimonie. En effet, les autres ont pour coefficient une valeur fixe donnée par le problème : celle de la borne qui les contraint (0 et C dans le cas des C-SVM par exemple). Partant de ce constat, il suffirait de connaître la répartition des points dans trois groupes (contraints à 0, contraints à C et non contraints) pour avoir très rapidement (en une résolution de système linéaire sur les points non contraints) la solution du problème. On appellera par la suite les groupes I_0, I_C et I_w, respectivement pour les points contraints à 0, à C et non contraints. Ce constat a amené à différentes techniques de décompositions et différents algorithmes que nous allons détailler après avoir donné quelques outils communs à plusieurs méthodes.

2.2.3 Les outils utilisés

Directions admissibles La géométrie du problème quadratique convexe dual est résumée par la figure 2.1 page suivante. Les contraintes de boîte $0 \leq \alpha_i \leq C$ restreignent la solution à un hypercube de dimension n. La contrainte d'égalité $\sum y_i\alpha_i = 0$ restreint en plus la solution à un polytope $\mathscr{F}$ de dimension $n-1$. On considère un point admissible $\alpha \in \mathscr{F}$ et une droite contenant ce point. Cette droite indique une *direction admissible* si son intersection avec le polytope contient des points autres que α.

Les algorithmes de direction admissible mettent à jour de manière itérative la position d'un point admissible α_t. En premier lieu ils choisissent une direction $\mathbf{u}_t$ et ensuite ils suivent cette direction à la recherche du point admissible α_{t+1} qui maximise la fonction coût. L'optimum est atteint quand aucune amélioration n'est plus possible Zoutendijk [1960]. Il existe deux configurations possibles pour la recherche du point admissible. La fonction de coût quadratique restreinte à la direction de recherche atteint son maximum soit à l'intérieur, soit à l'extérieur du polytope.

Shrinking Le shrinking Joachims [1999] est une heuristique visant à déterminer au cours de l'avancement de l'algorithme quels points seront certainement exclus de la solution ou bornés. On connaîtra leur valeur sans avoir à calculer leur coefficient α. De cette façon, on peut ne plus tenir compte de ces points et réduire mécaniquement la taille du problème à résoudre. Comme cette heuristique est faillible, il faut vérifier à l'arrêt de l'algorithme que les points exclus sont dans le bon groupe I_0 ou I_C et éventuellement de refaire une étape d'optimisation.

2.2.4 Structure commune des algorithmes de résolution

Les algorithmes itératifs de résolution partagent une structure commune (algorithme 1 page suivante). Les méthodes se distinguent ensuite par leur façon de répartir les points et par le calcul des α.

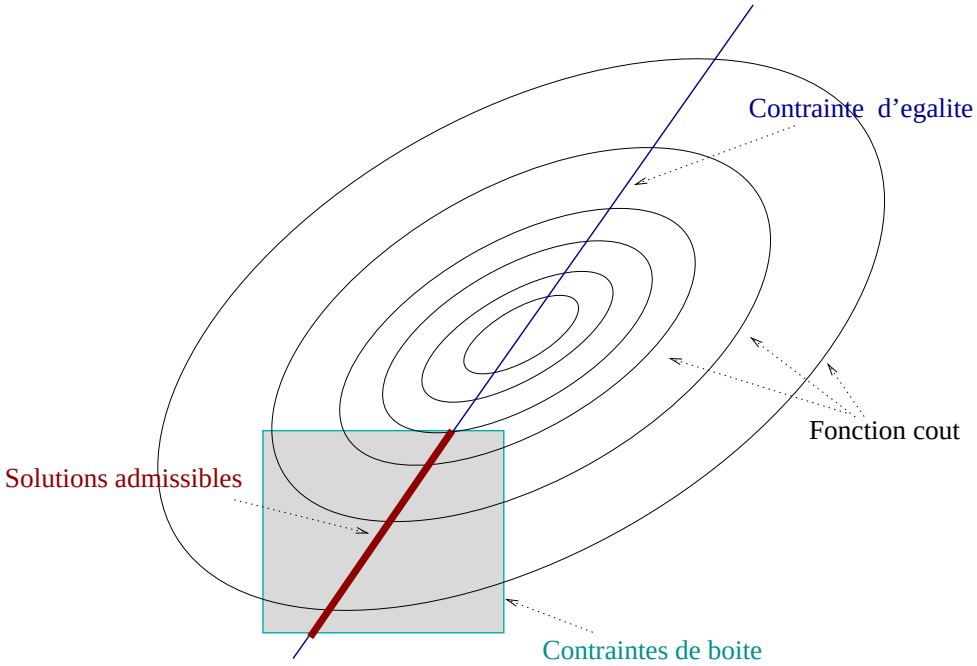


FIG. 2.1 : Géométrie du problème quadratique dual. Les contraintes de boîte $0 \leq \alpha_i \leq C$ restreignent la solution à un hypercube de dimension n. La contrainte d'égalité $\sum \mathbf{y}_i \alpha_i = 0$ restreint de plus la solution à un polytope de dimension $n-1$. La fonction de coût quadratique limitée à la ligne de recherche peut avoir son maximum à l'intérieur ou à l'extérieur de la boîte de contraintes.

Algorithme 1 Schéma général des algorithmes de résolution itératifs

1: initialisation
2: **tant que** la solution courante n'est pas optimale **faire**
3: mettre à jour à la répartition des groupes
4: calculer les coefficients α_i correspondant aux changements
5: **fin tant que**

2.2.5 Méthodes de décomposition

Les méthodes de décomposition, comme leur nom l'indique, utilisent des sous-ensembles de la base d'apprentissage à chaque étape, de manière à résoudre des problèmes de petite dimension. Les résultats de ces sous-problèmes sont ensuite combinés pour arriver à la solution optimale globale.

Chunking La méthode connue sous le nom de chunking Vapnik [1982] élimine les points dont les α valent 0 au fur et à mesure de l'avancement de l'algorithme. L'idée est de résoudre le problème QP pour un sous ensemble de points. Le sous ensemble suivant est composé des vecteurs supports de la résolution précédente et des M points sélectionnés dans le reste de la base comme étant ceux qui violent le plus les contraintes KKT. Ainsi, de sous groupe en sous groupe, on résout le problème en entier. La limite de cette méthode est la taille de la solution finale : on ne peut pas nécessairement stocker l'intégralité de la matrice noyau en mémoire.

Décomposition La technique de décomposition de Osuna et al. [1997] est similaire au chunking. Pour cette approche, on garde une taille de sous ensemble constante en laissant la possibilité de retirer du sous ensemble

courant des points vecteurs supports. Ainsi on peut résoudre les problèmes quadratiques quelque soit la taille réelle du problème.

SMO Le cas extrême de la décomposition consiste à ne considérer que deux points à chaque étape. Cette méthode est connue sous le nom SMO (Sequential Minimal Optimisation) Platt [1999]. L'intérêt principal est que chaque sous problème quadratique a une taille suffisamment restreinte pour être résolu analytiquement, évitant ainsi une résolution numérique coûteuse.

SMO ne considère que les directions admissibles qui ne modifient que deux coefficients α_i et α_j par des valeurs opposées. La variante la plus répandue du SMO repose sur un critère du premier ordre pour sélectionner les paires (i, j) qui définissent les directions successives de recherche :

$$i = \underset{\{s \mid \alpha_s < \max(0, y_s C)\}}{arg\,max} \frac{\partial W}{\partial \alpha_s} \qquad j = \underset{\{s \mid \alpha_s > \min(0, y_s C)\}}{arg\,min} \frac{\partial W}{\partial \alpha_s} \tag{2.8}$$

Par ailleurs, la plupart des implémentations de SMO (schéma itératif donné par l'algorithme 2) font appel à la

Algorithme 2 Schéma itératif de l'algorithme SMO

1: choisir deux points i et j initiaux
2: calculer leur coefficients α_i et α_j
3: **tant que** la solution courante n'est pas optimale **faire**
4: sélectionner deux nouveaux points
5: calculer les coefficients associés
6: **fin tant que**

technique de shrinking de manière à restreindre l'espace de recherche des α. Un nouveau critère de sélection basé sur l'information de second ordre a été publié récemment [Rong-En Fan, 2005] et assure des résultats aussi bien théoriques que pratiques sur une meilleure convergence.

L'utilisation de SMO permet de se dégager de l'utilisation de solveurs QP. Pour les autres méthodes de décomposition, il reste pour chaque sous problème à résoudre un problème quadratique.

2.2.6 Résolution d'un problème quadratique

Méthodes de point intérieur Les méthodes de point intérieur, proposées par Karmarkar [1984], sont le fruit de recherches pour résoudre la programmation linéaire en temps polynomial. Leur dérivation pour la programmation semi-définie positive et non linéaire a donné les algorithmes de type LOQO et OOQP [Gertz and Wright, 2001, Vanderbei, 1999]. Ces méthodes, bien qu'à la pointe de la recherche en optimisation et très efficaces, sont toutefois surpassées dans le cas considéré par les méthodes que nous présentons ensuite. En effet, les méthodes de points intérieurs requièrent de partir de la base d'apprentissage complète et par la même sont limitées pour le cas des grandes bases de données, indépendamment de l'efficacité de l'algorithme ensuite. La particularité des problèmes rencontrés en apprentissage est la parcimonie, cette propriété est le point clef de la résolution des grands problèmes quadratiques. Même en travaillant sur un sous ensemble de points de la base d'apprentissage comme c'est le cas pour les méthodes de décomposition, on gagne en temps et en performance à faire appel à des méthodes qui utilisent la parcimonie pour réduire leur complexité.

Méthodes de contraintes actives Le principe des méthodes de contraintes actives est de répartir le plus efficacement possible les points dans les trois groupes I_0, I_C et I_w et de trouver les valeurs des $\alpha_i, i \in I_w$ par la résolution d'un système linéaire. Pour les SVM, l'algorithme est baptisé SimpleSVM [Vishwanathan et al., 2003] (toutefois il s'applique à tout problème posé sous la forme quadratique convexe sous contraintes de boîtes) et permet de trouver itérativement la répartition en trois groupes.

A chaque étape, on résout pour le groupe I_w la minimisation sans contrainte. Si une des valeurs de α ainsi calculées viole les contraintes (c'est-à-dire que la solution trouvée ne se situe plus dans la boîte de contraintes) alors on projette α dans l'espace admissible (ce qui revient en pratique à changer de groupe le point indicé par la valeur qui viole les contraintes).

Le schéma itératif de ce type d'algorithme est le suivant :

Algorithme 3 Schéma itératif de l'algorithme SimpleSVM

1: choisir une répartition initiale
2: calculer une première solution
3: **tant que** la solution courante n'est pas optimale **faire**
4: mettre à jour à la répartition des groupes
5: calculer les coefficients associés au groupe I_w
6: **fin tant que**

Il reste donc à définir comment choisir la répartition des groupes, comment calculer les coefficients et comment évaluer la performance de la solution ainsi trouvée. Les détails sont présentés dans le chapitre 3 page 35

2.3 Méthodes alternatives non optimales

La taille et la forme de certains problèmes pratiques ont conduit au développement de méthodes résolution non exactes mais qui présentent des avantages calculatoires ou structurels. Nous présentons ici deux exemples de ces méthodes approchées. Le premier, CVM (*Core Vector Machine*) [Tsang et al., 2005] est une méthode proche des méthodes de décomposition conçue pour les grandes bases de données et le second, LASVM [Bordes et al., 2005b] est une méthode développée pour l'apprentissage en ligne.

2.3.1 CVM

L'algorithme nommé *Core Vector Machine*, CVM, est une méthode pour laquelle est proposée une heuristique permettant d'obtenir une solution approchée en économisant du temps de calcul. Sans cette heuristique, CVM peut aussi donner une solution exacte mais ne présente alors pas d'intérêt calculatoire.

CVM utilise la notion de *Minimum Enclosing Ball*, MEB.

DÉFINITION - MEB : Le problème posé par le MEB est de trouver la sphère de rayon minimum contenant un ensemble de points donnés. Une méthode de résolution consiste à choisir deux points au hasard et d'initialiser le premier centre à mi-chemin des deux points et le premier rayon à la distance entre le centre et chaque point. Ensuite on modifie le centre et le rayon de manière à englober le point se trouvant le plus loin du centre existant. On procède ainsi tant qu'il reste des points hors de la sphère.

CVM utilise une ε-sphère, c'est-à-dire une sphère qui couvre l'ensemble des points à ε près. On note c le centre de la sphère et R le rayon. Le problème que l'on cherche à résoudre est peut donc se formuler comme suit :

$$\min_{c,R} \|c - \phi(\mathbf{x}_i)\|^2 \leq R^2 \quad \forall i_i n[1,...,n]$$

Le problème dual est :

$$\begin{cases} \max\limits_{\alpha} & \alpha' diag(K) - \alpha' K \alpha \\ & \alpha \geq \mathbf{0} \\ & \alpha' \mathbf{1} = 1 \end{cases}$$

Pour un noyau gaussien, ou un noyau normalisé ou encore un noyau polynomial avec entrée normalisées, il est possible de simplifier ce problème car $\alpha' diag(K) = cst$:

$$\begin{cases} \max\limits_{\alpha} & -\alpha' K \alpha \\ & \alpha \geq \mathbf{0} \\ & \alpha' \mathbf{1} = 1 \end{cases}$$

Le L2-SVM (les variables d'écart ξ_i sont au carré) peut être écrit sous cette forme si le noyau contient les informations suivantes : $\tilde{k(\mathbf{x}_i, \mathbf{x}_j)} = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + y_i y_j + \frac{\delta_{ij}}{C}$ avec $\delta_{ij} = 1$ si $i = j$ et 0 sinon.

CVM est dans son déroulement très proche des méthodes de décompositions telles que SimpleSVM. En effet la méthode consiste en deux phases alternées, l'une qui définit les groupes de points et l'autre qui calcule les multiplicateurs de Lagrange pour un sous ensemble de points.

Définition des groupes Les points sont divisés en deux groupes de points. Il y a d'un côté l'ensemble I_{ncs} des points n'entrant pas dans la résolution du problème quadratique (l'ensemble des points à la première étape donc) et de l'autre côté ce qui est appelé le *coreset* et noté I_{cs} et qui rassemble les points d'intérêts. Ces points ne sont pas nécessairement des vecteurs supports et sont sélectionnés par la boucle externe de la méthode de résolution (cf. algorithme 4). Ce *coreset* est l'ensemble sur lequel est résolu le problème quadratique. Par conséquent, seuls les points de I_{cs} peuvent être vecteurs supports. Enfin, il faut noter qu'il n'existe pas dans cette méthode de phase de retrait de points de I_{cs}. Ainsi cet ensemble ne peut que grandir à chaque étape et ce jusqu'à convergence.

Calcul des multiplicateurs de Lagrange Les multiplicateurs de Lagrange sont par défaut fixés à 0. Pour les points du *coreset*, on résout un programme quadratique pour fixer les multiplicateurs correspondant. Pour cela, la méthode SMO est employée avec reprise à chaud d'une étape à l'autre (en effet, seul un point est incorporé à chaque étape et il est donc possible de mettre à jour la solution précédente). Cela constitue la boucle interne.

Algorithme 4 CVM

1: initialisation : deux points sont mis dans le *coreset* et définissent le premier centre de la boule et le premier rayon
2: **tant que** $\exists i \in I_{ncs}$ à l'extérieur de la boule **faire** ▷ boucle externe
3: sélectionner dans I_{ncs} le point le plus éloigné de la boule
4: résoudre le programme quadratique pour les points de I_{cs} avec SMO ▷ boucle interne
5: mettre à jour le centre et le rayon de la boule
6: **fin tant que**

Dans l'algorithme 4, deux étape requièrent un effort de calcul conséquent. D'une part le calcul de distance entre les points et la boule et d'autre part le SMO. Pour répondre à cela, il est proposé d'utiliser une heuristique

pour la boucle externe. Celle-ci consiste à tirer au hasard dans I_{ncs} 59 points et de sélectionner le prochain point entrant dans I_{cs} parmi ces 59 points. De cette façon, on assure avec 95% de confiance que le point sélectionné fait parti des 5% de points les plus éloignés du centre de la boule.

2.3.2 LASVM

Regarder du côté des méthodes en lignes est intéressant pour les application réalistes. En effet leur objectif est de combiner la performance des résultats des méthodes de type SVM à une utilisation à grande échelle et adaptative. LASVM est un algorithme qui combine les atouts de SMO et de SimpleSVM pour parvenir à cet objectif.

LASVM [Bordes et al., 2005b] est un SVM en ligne qui augmente incrémentalement la fonction objectif duale. Il maintient un vecteur des coefficients courants α et l'ensemble des indices des vecteurs supports correspondants I_v (ici, I_v est l'union de I_w et de I_C). Chaque itération de LASVM reçoit un nouvel exemple issu de I_0 et met à jour le vecteur de coefficients α en faisant deux étapes de SMO appelées *process* et *reprocess*.

- *Process* est une recherche de point admissible. La direction est définie par la paire de points formée de l'indice de l'exemple courant σ et d'un autre exemple choisi dans I_v en utilisant le critère de premier ordre (eq. 2.8 page 25). Cette opération donne un nouveau vecteur de coefficients α'_t et peut insérer σ dans l'ensemble d'indices I_v correspondant,
- *Reprocess* est aussi une recherche de point admissible mais la direction est cette fois définie par la paire de points (i, j) choisis tous les deux dans I_v en utilisant le critère de premier ordre (eq. 2.8 page 25). Cette opération donne un nouveau vecteur de coefficients α et peut enlever i ou j de l'ensemble des indices I_v.

Répéter les itérations LASVM (algorithme 5) sur des exemples choisis aléatoirement converge vers la solution SVM avec une précision arbitraire. Toutefois lorsque que n est grand, l'expérience montre que LASVM donne aussi de bons résultats après une seule présentation de chaque exemple. Après avoir présenté chaque exemple, les coefficients finaux sont affinés en faisant des *reprocess* jusqu'à convergence de la fonction duale.

Algorithme 5 Schéma de l'algorithme LASVM pour un fonctionnement en ligne.

```
initialiser
tant que il reste des points non vus faire
    sélectionner le prochain point
    Process
    Reprocess
fin tant que
finaliser
```

Chaque itération de LASVM en ligne sélectionne aléatoirement un exemple d'apprentissage $(\mathbf{x}_{\sigma(t)}, y_{\sigma(t)})$. Des stratégies plus sophistiquées de sélection d'exemples conduisent à de meilleurs performances de mise à l'échelle. On peut citer quatre stratégies :

- *sélection aléatoire* : Prendre un exemple non vu au hasard,
- *sélection par gradient* : Prendre l'exemple le moins bien classé (la plus petite valeur de $y_k f(\mathbf{x}_k)$) parmi un ensemble de points non vus. Ce critère est très proche de ce qui est fait

dans SimpleSVM,

– *sélection active* : Prendre l'exemple qui est le plus proche de la frontière de décision (la plus petite valeur de $\left|f(\mathbf{x}_k)\right|$) parmi un ensemble de poins non vus. Ce critère choisi un exemple indépendamment de son étiquette,

– *sélection autoactive* : Tirer au plus 100 points non vus, mais arrêter dès que 5 d'entre eux sont à l'intérieur des marges. Parmi les 5, choisir le plus proche de la frontière de décision.

On montre empiriquement que la sélection *active* ou *autoactive* donnent des performances comparables ou meilleures en utilisant un nombre plus restreint de vecteurs supports. Cela se comprend car l'accroissement linéaire du nombre de vecteurs supports est lié au fait que tout exemple mal classé est automatiquement vecteur support dans la formulation classique du SVM. Sélectionner les points uniquement près de la frontière exclue de la solution un grand nombre de points bruités.

De par sa structure, LASVM permet d'exploiter de grandes bases de données. En effet, un problème d'apprentissage n'est pas exactement un problème d'optimisation. En particulier cela n'a pas beaucoup de sens d'optimiser la fonction objectif primale avec une précision beaucoup plus importante que celle issue du fait de travailler avec un nombre fini d'exemples. Des résultats formels existent pour l'apprentissage en ligne, qui utilisent une approximation du gradient stochastique. Ces algorithmes conçus pour le fonctionnement en ligne vont beaucoup plus vite que l'optimisation directe de la fonction objectif primale [Bottou and LeCun, 2004] : ils n'optimisent pas cette fonction de manière aussi précise mais ils atteignent une erreur en test équivalente plus rapidement.

2.4 Résumé des méthodes de résolution

Nous allons résumer ici les points communs et les différences entre quelques-unes des principales méthodes de résolution des SVM, exactes ou approchées, à savoir SMO, SimpleSVM, LASVM, CVM.

Le tableau 2.2 page suivante résume les mécanisme de chaque méthode et les tableaux 2.3 page suivante, 2.4 page 31, 2.5 page 31 et 2.6 page 32 détaillent respectivement chaque étape, à savoir l'ajout d'un point dans la solution, le retrait d'un point, l'optimisation des multiplicateurs de Lagrange et les critères d'arrêt.

Conclusion

Nous avons, dans ce chapitre, donné un apperçu des méthodes utilisées pour résoudre des problèmes quadratiques sous contraintes tels que ceux qui apparaissent dans les SVM. Si le SMO est la méthode de résolution la plus populaire, les méthodes de contraintes actives peuvent apporter plusieurs avantages que nous allons présenter, via la description de SimpleSVM (chapitre 3) et la comparaison des comportements de diverses méthodes en fonction des conditions d'utilisation (chapitre 5).

Nous avons également présenté deux approches alternatives aux méthodes exactes. L'une est une méthode approchée, CVM et l'autre est une méthode en ligne, LASVM. L'étude de LASVM sera reprise dans le chapitre 6 lorsque nous aborderons le problème des invariances et des très grandes bases de données. En effet, l'apprentissage en ligne est un cadre idéal lorsque la taille des données interdit l'utilisation d'une technique qui réutilise plusieurs fois chaque exemple comme c'est le cas aussi bien pour SMO que pour SimpleSVM. L'étude de CVM est plus détaillée dans le chapitre 5. Dans ce chapitre nous nous intéresserons aux effets des crièteres d'arrêts des méthodes

	SimpleSVM	SMO

Algorithme global	– Initialisation (I_w)
	– Faire
	  – Phase d'optimisation des α_w
	  – Phase de suppression ou phase d'ajout
	– Tant que le critère d'arrêt n'est pas satisfait
		– Initialisation (i et j)
		– Faire
		  – Phase d'optimisation des α_i et α_j
		  – Phase de recherche d'un couple de points i et j
		– Tant que le critère d'arrêt n'est pas satisfait

	CVM	LASVM

Algorithme global	– Initialisation (coreset)
	– Faire
	  – Phase d'optimisation des $\alpha_{coreset}$
	  – Phase d'ajout d'un point dans le coreset
	– Tant que le critère d'arrêt n'est pas satisfait
		– Initialisation (I_{Cw})
		– Faire
		  – Phase d'ajout (i, j)
		  – Phase d'optimisation des α_i et α_j
		  – Phase de retrait (k, l)
		  – Phase d'optimisation des α_k et α_l
		– Tant que le critère d'arrêt n'est pas satisfait

TAB. 2.2 : Mécanique des algorithmes

	SimpleSVM	SMO	CVM	LASVM
Ajout d'un point	Cherche dans I_0 et dans I_C un point qui viole les contraintes (qui est du mauvais côté de sa marge) et transfert ce point vers I_w	L'ajout d'un point intervient quand le couple de points qui définit la meilleure direction de descente du gradient contient un point dont le multiplicateur associé était nul	Cherche hors du coreset un point qui viole les contraintes et transfert ce point dans le coreset.	Sélectionne un point parmi les données arrivantes selon un critère pré-défini (le plus près de la marge, le plus mal classé...) et le fixe comme un des deux points du couple définissant la direction de descente du gradient, l'autre point étant choisit dans les vecteurs supports.

TAB. 2.3 : Insertion d'un point dans la solution . Les principales différences sont d'une part l'ensemble dans lequel le point est choisi et d'autre part le critère de sélection

	SimpleSVM	SMO	CVM	LASVM
Retrait d'un point	Retire le point de I_w qui viole le plus les contraintes de boîte ($0 < \alpha_w < C$) pour le mettre dans le groupe correspondant à la contrainte violée (dans I_0 si le α concernné est négatif, dans I_C s'il est plus grand que C) et projette la solution courante dans une zone admissible	La suppression d'un point se fait mécaniquement quand pour un couple de points sélectionnés, l'un des multiplicateurs associés devient négatif, donc bloqué à 0 ou lorsqu'il devient plus grand que C et est alors bloqué également.	Aucun point n'est jamais enlevé du coreset. Au sein du coreset, les points sont ou non vecteurs supports selon le résultat donné par le SMO.	La suppression est la même que pour SMO quand les deux points sont sélectionnés parmi les vecteurs supports. Tout point dont le multiplicateur devient négatif est définitivement oublié.

TAB. 2.4 : Suppression d'un point de la solution courante . L'optimisation de la solution courante conduit à des éliminations de points s'ils ne sont plus vecteurs supports. L'optimisation peut toutefois ne pas être complète à chaque étape

	SimpleSVM	SMO	CVM	LASVM
Optimisation des multi-plicateurs α	Résolution d'un système linéaire où α est l'inconnu. A chaque étape la solution du système est mise à jour par une opératin de rang 1. Tous les multiplicateurs de la solution courante sont mis à jour en même temps.	Les multiplicateurs sont optimisés deux par deux pour que cela soit fait sans résolution de système. Les couples sont choisis selon une heuristique qui vise à sélectionner les points qui définissent la plus grande pente vers le minimum recherché.	Le programme quadratique sur le coreset est résolu par SMO, mis à jour à chaque étape (reprise à chaud).	Chaque étape d'optimisation de LASVM est une étape de SMO entre les deux points sélectionnés.

TAB. 2.5 : Calculs des multiplicateurs de Lagrange

	SimpleSVM	SMO	CVM	LASVM
Arrêt et finalisation	Arrêt des itérations quand tous les points sont bien classés (au sens de leur groupe I_0, I_C ou I_w) au saut de dualité près (*KKT gap*). Critère du C-SVM (eq. 5.3 page 64).	Arrêt des itérations quand il n'existe plus de direction de descente au saut de dualité près (*KKT gap*). Critère du C-SVM (eq. 5.3 page 64).	Dispose de deux critères d'arrêt. Le premier permet d'arrêter l'iteration de sélection de nouveaux points. Le deuxième est le critère d'arrêt de SMO qui intervient à chaque étape de CVM pour l'optimisation des $\alpha_{coreset}$. Critères de type ν-SVM (eq. 5.6 page 65)	Arrêt des itérations quand il n'y a plus de nouveau point à voir. La phase de finalisation qui suit consiste à enchaîner des étapes d'optimisation sur les points vecteurs supports jusqu'à respect du critère des C-SVM (eq. 5.3 page 64). Remarquons que cela n'est vérifié que pour les points qui ont été sélectionnés, les autres ont été oubliés définitivement.

TAB. 2.6 : Critères d'arrêt des méthodes de résolution . Nous revenons en détail sur l'impact des critères d'arrêt dans le chapitre 5

et aussi de l'effet de l'arrêt prématuré de l'apprentissage et montrerons qu'il est important de considérer avec prudence certaines approximations ainsi que les conditions d'arrêt des méthodes.

Deuxième partie

Outils et contributions

3

SimpleSVM [1]

« Une méthode fixe n'est pas une méthode. »

Proverbe chinois

Sommaire

3.1 Relation entre primal et dual 35
3.2 Structure et détails de SimpleSVM 37
3.3 Variations sur le même thème 40

UNE MÉTHODE DE RÉSOLUTION du problème des SVM est présentée dans ce chapitre. Basé sur le principe des contraintes actives, SimpleSVM permet de résoudre efficacement un problème quadratique sous contraintes et est riche en extensions potentielles. Les différentes étapes de résolution sont étudiées et justifiées puis nous donnerons quelques exemples d'extensions et dérivations.

Les dérivations des SVM sont en général l'application du principe des SVM à des entrées ou à des sorties différentes du cas classique. C'est le cas pour le SVM à une classe (OC-SVM) pour lequel les étiquettes de la base d'apprentissage sont toutes identiques (il n'y a qu'une seule classe) et dont l'objectif est de determiner si un exemple nouveau fait parti ou non de la classe apprise.

Nous étudierons la reprise à chaud des calculs, qui permet entre autres d'accélerer les techniques de réglage des hyper-paramètres telle que la validation croisée ou le *leave-one-out*.

3.1 Relation entre primal et dual

Nous avons donné dans le chapitre 2 page 17 les formes primales et duales de SVM binaires à marge douce, ou encore C-SVM. Nous rappelons ici ces deux formes pour faire apparaître les équivalences entre les paramètres des deux formulations.

[1] Les travaux contenus dans ce chapitre ont donné lieu aux publications [Loosli et al., 2004, 2005b]

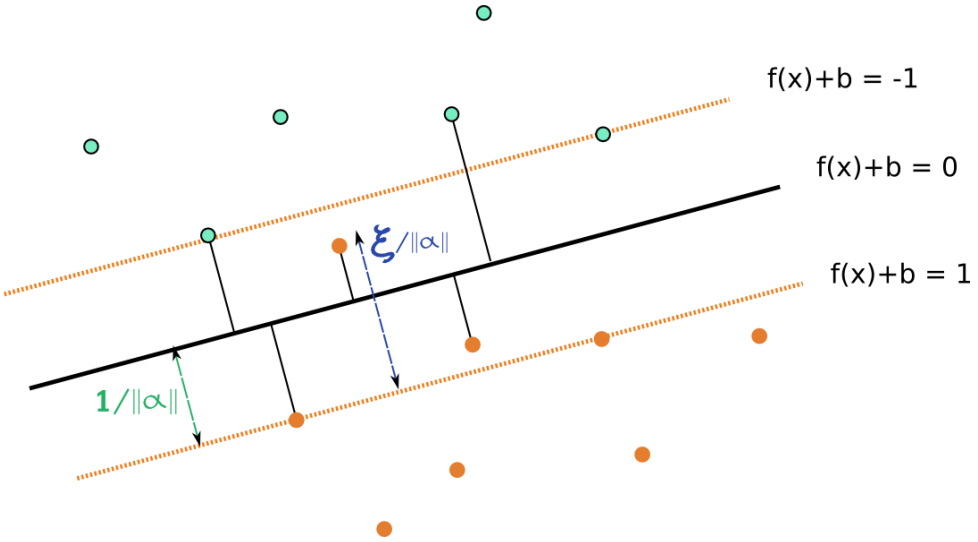


FIG. 3.1 : Illustration du SVM

Forme primale :

$$\begin{cases} \min\limits_{f\in\mathscr{H},\xi,b} & \frac{1}{2}\|f\|^2_{\mathscr{H}} + C\sum\limits_{i=1}^{n}\xi_i \\ & y_i(f(\mathbf{x}_i)+b) \geq 1-\xi_i \quad i\in[1,...,n] \\ & \xi_i \geq 0 \qquad\qquad i\in[1,...,n] \end{cases} \tag{3.1}$$

Les valeurs à identifier ici sont les variables d'écart ξ et le biais b. La forme duale s'écrit :

$$\begin{cases} \max\limits_{\alpha\in\mathbb{R}^n} & -\frac{1}{2}\alpha' G\alpha + \mathbf{1}'\alpha \\ & \mathbf{y}'\alpha = 0 \\ & 0 \leq \alpha_i \leq C \qquad i\in[1,...,n] \end{cases} \tag{3.2}$$

Pour résoudre le problème dual, le Lagrangien est encore une fois utilisé :

$$\mathscr{L}(\alpha,\mu,\eta,\vartheta) = -\frac{1}{2}\alpha' G\alpha + \mathbf{1}'\alpha - \mu(\mathbf{y}'\alpha) - \eta'(\alpha - C\mathbf{1}) + \vartheta'\alpha$$

avec μ, η et ϑ des multiplicateurs de Lagrange positifs.

Pour faire apparaître les équivalences entre les multiplicateurs de Lagrange et les variables du problème primal, les trois groupes I_w, I_0 et I_C sont considérés successivement et explicités dans le Lagrangien :

$$\begin{aligned} \mathscr{L}(\alpha_w,\alpha_C,\mu,\eta_w,\eta_0,\vartheta_w,\vartheta_C) \quad = \quad & -\frac{1}{2}\alpha'_w G_{(w,w)}\alpha_w - \frac{1}{2}\alpha'_C G_{(C,C)}\alpha_C - \alpha'_C G_{(C,w)}\alpha_w && (3.3)\\ & +\mathbf{1}'_w\alpha_w + \mathbf{1}'_C\alpha_C && (3.4)\\ & -\mu(\mathbf{y}'_w\alpha_w + \mathbf{y}'_C\alpha_C) && (3.5)\\ & -\eta'_w(\alpha_w - C\mathbf{1}_w) + \eta'_0(C\mathbf{1}_0) && (3.6)\\ & +\vartheta'_w\alpha_w + \vartheta'_C\alpha_C && (3.7) \end{aligned}$$

Points non contraints : I_w. Si on ne considère que les points du groupe I_w, tous ont un multiplicateur α compris entre 0 et 1. Les multiplicateurs η_w et ϑ_w correspondants aux points non contraints sont nuls puisque les contraintes

de boîte ne sont pas actives. L'annulation du gradient de (3.3) par rapport à α_w donne $G_{(w,.)}\alpha + \mu \mathbf{y}_w - \mathbf{1}_w = \mathbf{0}_w$. Par ailleurs, dans l'espace primal, les points non contraints sont situés sur la marge, c'est-à-dire que $y_i(f(\mathbf{x}_i)+b)-1 = 0$. Sous forme vectorielle pour tous les points de I_w, cela équivaut à $G_{(w,.)}\alpha + b\mathbf{y}_w - \mathbf{1}_w = \mathbf{0}_w$. Par identification, on obtient la relation $b = \mu$.

Points contraints à C : I_C. Les points de I_C sont tous bornés à C, ainsi seule la contrainte supérieure de boîte est active, $\vartheta_C = 0$ et $\eta_C > 0$. L'annulation du gradient d (3.3) par rapport à α_C donne $-G_{(C,.)}\alpha - \mu \mathbf{y}_C + \mathbf{1}_C = \eta_C$. Par ailleurs, dans l'espace primal, les points contraints à la borne C sont situés du mauvais côté de la marge par rapport à leur classe et les variables d'écart associées ξ_C sont strictement positives, c'est-à-dire que $y_i(f(\mathbf{x}_i)+b)-1 = -\xi_i$. Sous forme vectorielle pour tous les points de I_C, cela équivaut à $G_{(C,.)}\alpha + b\mathbf{y}_C - \mathbf{1}_C = -\xi_C$. Par identification, on obtient la relation $\xi_C = \eta_C$. Par ailleurs, ces valeurs sont nulles pour les autres groupes de points car la contrainte de boîte à C est inactive et par conséquent, $\xi = \eta$.

Points contraints à 0 : I_0. Les points de I_0 sont tous bornés à 0, ainsi seule la contrainte inférieure de boîte est active, $\vartheta_0 > 0$ et $\eta_0 = 0$. L'annulation du gradient du Lagrangien par rapport à α_0 donne $G_{(0,.)}\alpha + \mu \mathbf{y}_0 - \mathbf{1}_0 = \vartheta_0 > 0$. Par ailleurs, dans l'espace primal, les points contraints à la borne 0 sont situés du bon côté de la marge par rapport à leur classe, c'est-à-dire que $y_i(f(\mathbf{x}_i)+b)-1 > 0$. Sous forme vectorielle pour tous les points de I_0, cela équivaut à $G_{(0,.)}\alpha + b\mathbf{y}_0 - \mathbf{1}_0 > 0$.

Récapitulatif Ces relations sont résumées dans le tableau 3.1. Vérifier ces conditions d'optimalité revient à calculer $G\alpha + b\mathbf{y} - \mathbf{1}$ puis à s'assurer que pour tous les points de I_0 ($\alpha = \mathbf{0}$) cette quantité soit positive et que pour tous les points de I_C ($\alpha = C$) cette quantité soit bien négative.

Ensemble	Contraintes initiales	Contraintes primales		Contraintes Duales	
I_w	$y_i(f(\mathbf{x}_i)+b) = 1$	$0 < \alpha < C$,	$\xi = 0$	$\vartheta_w = 0$,	$\eta_w = 0$
I_C	$y_i(f(\mathbf{x}_i)+b) = 1-\xi_i$	$\alpha = C$,	$\xi > 0$	$\vartheta_C = 0$,	$\eta_C > 0$
I_0	$y_i(f(\mathbf{x}_i)+b) > 1$	$\alpha = 0$,	$\xi = 0$	$\vartheta_0 > 0$,	$\eta_0 = 0$

TAB. 3.1 : Situation des contraintes pour les trois types de variables.

Pour résoudre efficacement le problème, il est astucieux de prendre en compte la situation des points au regard des contraintes vérifiées ou non.

3.2 Structure et détails de SimpleSVM

L'objectif de tout algorithme SVM est double : il faut d'une part arriver à répartir l'ensemble d'apprentissage dans ces trois groupes, puis une fois la répartition connue, résoudre le problème. Il s'avère que cette seconde phase est relativement plus facile.

Supposons que l'on connaisse la répartition des points (I_w, I_0 et I_C donnés) : les contraintes d'inégalité ne sont plus nécessaires (elles sont contenues implicitement dans la définition des trois ensembles de points). Seules les valeurs des α_i pour $i \in I_w$ demeurent inconnues, car par définition $\alpha_i = 0$ pour $i \in I_0$ et $\alpha_i = C$ pour $i \in I_C$. La valeur

des α_i pour $i \in I_w$ est ainsi donnée par la solution du problème d'optimisation suivant :

$$\left\{ \begin{array}{ll} \max\limits_{\alpha_w \in \mathbb{R}^{|I_w|}} & -\frac{1}{2}\alpha'_w G_{(w,w)}\alpha_w + (\mathbf{1}'_w - C\mathbf{1}'_C G_{(C,w)})\alpha_w \\ \text{avec} & \alpha'_w \mathbf{y}_w + C\mathbf{1}'_C \mathbf{y}_C = 0 \end{array} \right. \tag{3.8}$$

Il est ici important de noter que la dimension de ce problème est égale au cardinal de l'ensemble I_w (qui peut être bien inférieur à n, la dimension initiale du problème). Les conditions de Kuhn Kurush et Tucker nous donnent le système à résoudre pour obtenir la valeur des coefficients α_w encore inconnus :

$$\left(\begin{array}{cc} G_{(w,w)} & \mathbf{y}_w \\ \mathbf{y}'_w & 0 \end{array} \right) \left(\begin{array}{c} \alpha \\ \mu \end{array} \right) = \left(\begin{array}{c} \mathbf{1}_w - CG_{(w,C)}\mathbf{1}_C \\ -C\mathbf{1}'_C \mathbf{y}_C \end{array} \right) \tag{3.9}$$

Si la solution de ce système contient une composante violant les contraintes (une composante négative ou supérieure à C), elle prouve que la répartition initiale des points I_w, I_0 et I_C est fausse. Cette composante violant les contraintes doit être enlevée de I_w pour passer dans I_0 ou I_C. Si maintenant toutes les composantes de la solution α vérifient les contraintes, il n'est pas encore certain que nous ayons trouvé l'optimum du problème initial. Il faut encore vérifier que les contraintes de positivité sont respectées pour les multiplicateurs de Lagrange associés. Alors, nous allons vérifier que pour tous les points de I_0, $G_{(0,.)}\alpha + b\mathbf{y}_0 - \mathbf{1}_0 > \mathbf{0}$, et que pour tous les points de I_C, $G_{(C,.)}\alpha + b\mathbf{y}_C - \mathbf{1} < \mathbf{0}$. Si tel n'est pas le cas, le point violant le plus les contraintes devra changer d'appartenance et quitter I_0 ou I_C pour passer dans I_w. Nous venons là de donner le principe de notre algorithme. Il s'agit d'un algorithme itératif qui va, à chaque itération, ajouter ou enlever un seul point de l'ensemble I_w. Nous verrons qu'à chaque itération le coût décroît strictement, garantissant ainsi la convergence de la méthode. Enfin, puisqu'entre chaque itération les matrices $G_{(w,w)}$ ne diffèrent que d'une ligne et d'une colonne, il est possible de calculer la nouvelle solution directement à partir de l'ancienne, réduisant ainsi la complexité de chaque itération de $\mathcal{O}(|I_w|^3)$ à $\mathcal{O}(|I_w|^2)$.

Une autre manière d'introduire l'algorithme est de le présenter comme une méthode de gradient projeté. Le principe consiste à partir de l'intérieur du domaine des contraintes admissibles et d'y rester en recherchant à chaque itération une direction de descente admissible qui nous amène sur la frontière du domaine. Une telle direction peut être obtenue en projetant le gradient sur cette frontière (Minoux [1983] page 197). C'est cette projection qui va nous amener à faire passer un point de I_w vers I_0 ou I_C.

3.2.1 Activation d'une contrainte : projection

Il faut ici faire attention de ne pas mélanger contrainte active et point actif dans la solution. Si l'on active une contrainte, cela veut dire que l'on fixe α à l'une des bornes (0 ou C) et par conséquent que le point correspondant est inactif dans le classifieur. Supposons que l'on connaisse un point admissible $\tilde{\alpha}_{\text{old}} = (\alpha_{\text{old}}, \mu_{\text{old}})$. L'optimum non contraint $\tilde{\alpha}^*$ est donné par l'équation (3.9) :

$$\tilde{\alpha}^* = H^{-1}\tilde{e} \qquad \text{avec} \quad H = \left(\begin{array}{cc} G_{(w,w)} & \mathbf{y}_w \\ \mathbf{y}'_w & 0 \end{array} \right) \quad \text{et} \quad \tilde{e} = \left(\begin{array}{c} \mathbf{1}_w - CG_{(w,C)}\mathbf{1}_C \\ -C\mathbf{1}'_C \mathbf{y}_C \end{array} \right) \tag{3.10}$$

On choisit la direction de descente $\mathbf{d} = \tilde{\alpha}^* - \tilde{\alpha}_{\text{old}}$, celle du gradient, qui garantit que le coût va décroître. Afin de rester dans le domaine admissible $0 \leq \alpha \leq C$, nous allons recenser toutes les composantes de α^* violant les contraintes, et rechercher le pas de descente t^* minimisant le coût tout en nous maintenant dans le domaine admissible, c'est-à-dire :

$$t^* = \max_t (\alpha = \alpha_{\text{old}} + t\mathbf{d} \,|\, 0 \leq \alpha \leq C) \tag{3.11}$$

Pratiquement, le pas de descente est donné directement par la recherche du minimum suivant :

$$t^* = \min_t \left(-\frac{\alpha_{\text{old}}}{\mathbf{d}}, \frac{C - \alpha_{\text{old}}}{\mathbf{d}} \right) \tag{3.12}$$

Cela revient à projeter sur la frontière de la boîte et à éliminer de l'ensemble I_w la variable correspondant au plus grand pas possible. La nouvelle solution est alors :

$$\tilde{\alpha}_{\text{new}} = \tilde{\alpha}_{\text{old}} + t^* \mathbf{d} \tag{3.13}$$

LEMME - DÉCROISANCE DU COÛT : Le coût diminue strictement à chaque itération (algorithme 6 page suivante).

PREUVE : Considérons le coût :

$$\mathscr{C}(\alpha) = \tfrac{1}{2}\alpha' H \alpha - \alpha' \tilde{e}$$

la direction de descente $\mathbf{d}$ est celle du gradient, elle vérifie :

$$\begin{aligned} \mathbf{d} &= \tilde{\alpha}^* - \tilde{\alpha}_{\text{old}} \\ &= H^{-1}\tilde{e} - \tilde{\alpha}_{\text{old}} \end{aligned}$$

et donc $H\mathbf{d} = \tilde{e} - H\tilde{\alpha}_{\text{old}}$. On a alors :

$$\begin{aligned} \mathscr{C}(\tilde{\alpha}_{\text{new}}) - \mathscr{C}(\tilde{\alpha}_{\text{old}}) &= \mathscr{C}(\tilde{\alpha}_{\text{old}} + t\mathbf{d}) - \mathscr{C}(\tilde{\alpha}_{\text{old}}) \\ &= \tfrac{1}{2}t^2 \mathbf{d}' H \mathbf{d} + t(H\tilde{\alpha}_{\text{old}} - \tilde{e})'\mathbf{d} \\ &= \tfrac{1}{2}t^2 \mathbf{d}' H \mathbf{d} - t\mathbf{d}' H \mathbf{d} \\ &= \left(\tfrac{1}{2}t^2 - t\right) \mathbf{d}' H \mathbf{d} \qquad < 0 \end{aligned}$$

car le premier terme est négatif pour $0 < t < 2$ et par définition, le pas est compris entre 0 et 1. Par ailleurs le second est positif puisque la matrice H est définie positive et que le vecteur $\mathbf{d}$ n'a pas toutes ses composantes nulles sauf la dernière. □

3.2.2 Désactivation d'une contrainte

Désactiver une contrainte signifie permettre au α concerné de prendre une valeur libre. Cela revient à faire passer le vecteur correspondant dans le groupe I_w, des vecteurs supports candidats.

Cela s'avère indispensable lorsque le vecteur $\tilde{\alpha}^*$ calculé précédemment est admissible (lorsque $t = 1$). Dans ce cas, pour qu'il existe encore une direction de descente (pour que le coût continue à décroître) il faut trouver soit un point de I_0 pour lequel $y_i(f(\mathbf{x}_i) + b) - 1 < 0$, soit un point de I_C pour lequel $y_i(f(\mathbf{x}_i) + b) - 1 > 0$. Ces contraintes correspondent au fait que le gradient est convexe au point concerné et qu'il est possible de diminuer le coût en entrant dans le domaine admissible.

De manière intuitive, on vérifie dans le primal que la solution candidate classe bien les points du jeu d'apprentissage. Si ce n'est pas le cas, alors il faut réajuster la solution en ajoutant au groupe des actifs un des points mal « classé » (au sens de I_w, I_0 et I_C).

3.2.3 Convergence

L'algorithme (6) résume le principe des itérations de l'algorithme du *simpleSVM*. Nous allons reprendre la preuve de convergence de l'algorithme donnée dans Vishwanathan et al. [2003].

Algorithme 6 : *simpleSVM*

1: $(I_w, I_0, I_C) \leftarrow$ initialiser
2: $(\alpha, \mu) \leftarrow$ resoudre le systeme sans contraintes(I_w) ▷ équation (3.10)
3: **tant que** le solution n'est pas optimale **faire**
4: **si** $\min \alpha_w < 0$ ou $\max \alpha_w > C$ **alors**
5: projeter α à l'intérieur du domaine admissible ▷ équation (3.13)
6: transférer le point associé de I_w vers I_0 ou I_C
7: **sinon**
8: calculer les multiplicateurs de Lagrange η_C et ϑ_0 ▷ tableau 3.1 page 37
9: **si** $\min \vartheta_0 < \varepsilon$ ou $\min \eta_C < \varepsilon$ **alors**
10: transférer un point associé de I_0 ou I_C vers I_w
11: **fin si**
12: **fin si**
13: **fin tant que**

L'algorithme s'arrête lorsque le vecteur $\tilde{\alpha}^*$ calculé précédemment est admissible et tous les points de I_0 et I_C vérifient leurs contraintes. Il n'existe alors plus de direction de descente admissible. Comme la solution dépend de la répartition des points dans les ensembles I_w, I_0 et I_C, qu'il n'existe qu'un nombre fini de points donc de combinaisons et que le coût associé à l'algorithme décroît strictement à chaque itération, l'algorithme ne peut pas boucler et va atteindre la solution globale en un temps fini.

3.3 Variations sur le même thème

Le SimpleSVM permet de résoudre tout le problème prennant la même forme. Ainsi, dans le cadre des SVM, on peut par exemple résoudre les SVM à une class ou encore le ν-SVM. Par ailleurs, sa structure permet de dériver d'autres méthodes (en effectuant des traitements particuliers au point sélectionné pour entrer dans l'ensemble I_w, comme c'est le cas pour les invariances (voir le chapitre 6 page 71) ou encore en tirant parti de la possibilité de reprise à chaud de la méthode (voir section vrefsec :reprise)).

3.3.1 Le SVM à une classe : OC-SVM

Le SVM à une classe, nommé OC-SVM pour *One-Class SVM*, possède la particularité de caractériser une classe plutôt que de discriminer deux classes. Cette approche est utilisée par exemple quand les données sont en très grande majorité d'une classe et que les contre-exemples sont rares ou inexistants. On n'apprend alors qu'à partir des données d'une seule classe. Les applications se trouvent dans la détection d'erreur ou de nouvelle classe (voir la *Novelty detection* dans Schölkopf and Smola [2002]).

La forme primale de ce problème est :

$$\left\{ \begin{array}{ll} \min\limits_{f\in\mathscr{H},\xi,b} & \frac{1}{2}\|f\|^2_{\mathscr{H}} + C\sum\limits_{i=1}^{n}\xi_i - \rho \\ & f(\mathbf{x}_i) \geq \rho - \xi_i \qquad i \in [1,...,n] \\ & \xi_i \geq 0 \qquad i \in [1,...,n] \\ & \rho \geq 0 \end{array} \right. \tag{3.14}$$

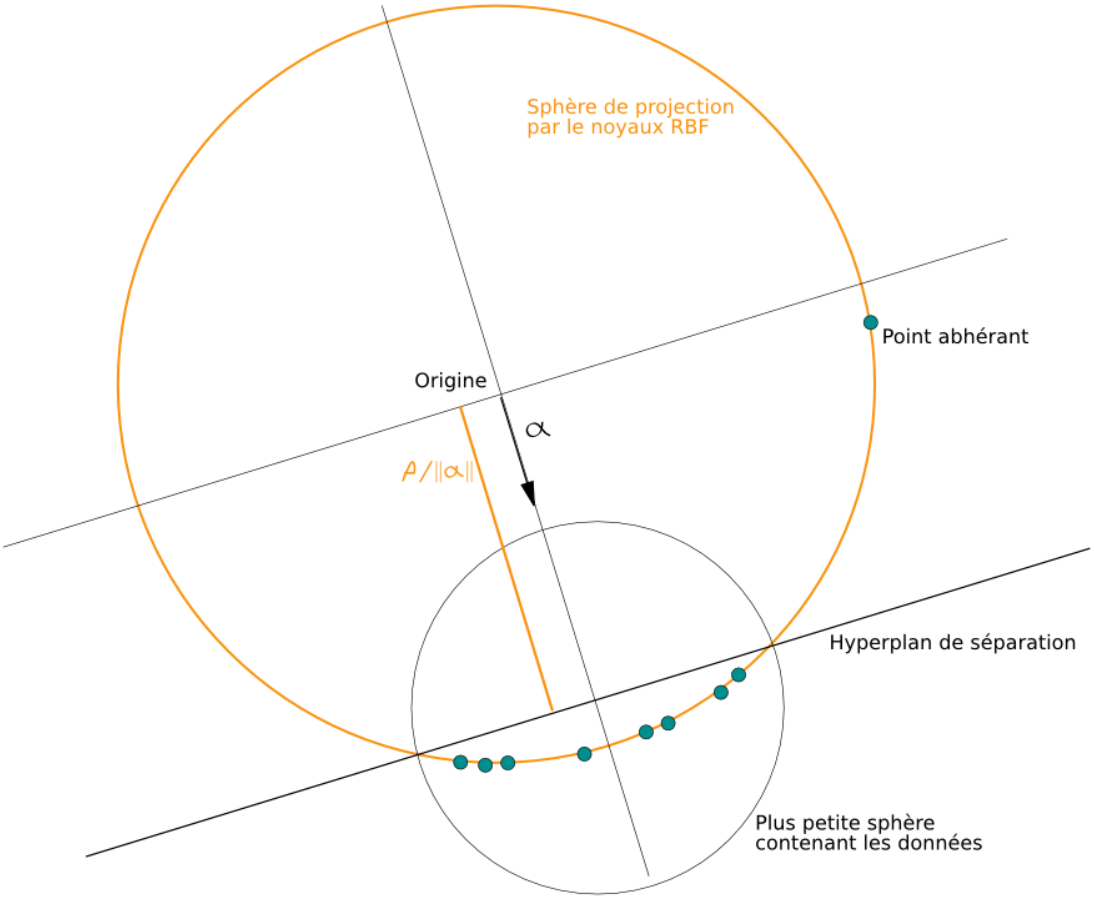


FIG. 3.2 : Illustration du fonctionnement du SVM à une classe. Pour un noyau de type gaussien ou RBF, les données sont projetées sur un sphère. Le SVM à une classe cherche à construire l'hyperplan tel que les données soit englobées dans un sphère de plus petit rayon possible. En relâchant les contraintes, on permet d'exclure de la sphère certains points qui sont alors considérés comme *outliers*, c'est-à-dire des points d'une autre classe ou points aberrants.

En passant par le Lagrangien, on aboutit à la forme duale :

$$\begin{cases} \max\limits_{\alpha\in\mathbb{R}^n} & -\frac{1}{2}\alpha' K\alpha \\ & \mathbf{1}'\alpha = 0 \\ & 0 \leq \alpha_i \leq C \quad i \in [1,...,n] \end{cases} \tag{3.15}$$

Pour résoudre le problème dual, le Lagrangien est utilisé :

$$\mathscr{L}(\alpha,\mu,\eta,\vartheta) = -\frac{1}{2}\alpha' K\alpha - \mu(\mathbf{1}'\alpha) - \eta'(\alpha - C\mathbf{1}) + \vartheta'\alpha$$

avec μ, η et ϑ des multiplicateurs de Lagrange positifs.

En utilisant la même technique de décomposition selon les groupes que précédemment et en dérivant le Lagrangien par rapport à α_w et à μ, on obtient les relations suivantes :

$$\alpha_w = K_{(w,w)}^{-1}(-CK_{(w,C)}\mathbf{1}_C + \mu\mathbf{1}_w) \tag{3.16}$$

$$\mu = \frac{1 - C|I_C| + C\mathbf{1}_C' K_{(C,w)} K_{(w,w)}^{-1}\mathbf{1}_w}{\mathbf{1}_w' K_{(w,w)}^{-1}\mathbf{1}_w} = \rho \tag{3.17}$$

L'algorithme de résolution est alors le même que le SimpleSVM, alternant le calcul de α_w et μ pour la répartition courante et le changement de répartition suivant la performance de la solution obtenue.

3.3.2 Le ν-SVM

Dans le SVM binaire classique à marges douces (le C-SVM), le paramètre de relâchement de contraintes C prend sa valeur dans $\mathbb{R}$ et cette valeur n'est pas interprétable directement. Le ν-SVM reformule le problème de façon à proposer un paramètre de relâchement ν variant entre 0 et 1. L'intérêt de cette reformulation réside dans le fait de pouvoir donner une interprétation *a priori* de ce paramètre. Il représente une proportion de points, d'une part la proportion minimum de points vecteurs supports et d'autre part la proportion maximum de points dont le multiplicateur de Lagrange associé est saturé.

Le problème primal du ν-SVM est le suivant :

$$\left\{\begin{array}{lll} & \min\limits_{f,b,\rho,\xi_i} \frac{1}{2}\|f\|^2 - \nu\rho + \frac{1}{n}\sum\limits_{i=1}^{n}\xi_i & \\ \text{avec} & y_i\big(f(x_i)+b\big) \geq \rho - \xi_i & \forall i \in [1,...,n] \\ \text{et} & \rho \geq 0 & \\ \text{et} & \xi_i \geq 0 & \forall i \in [1,...,n] \end{array}\right. \tag{3.18}$$

Le problème dual associé au problème primal (3.18) est :

$$\left\{\begin{array}{ll} \max\limits_{\alpha} -\frac{1}{2}\alpha' G\alpha & \\ \text{avec} \quad \alpha'\mathbf{1} \geq \nu & \\ \text{et} \quad \alpha'\mathbf{y} = 0 & \\ \text{et} \quad 0 \leq \alpha_i \leq \frac{1}{n} & \forall i \in [1,...,n] \end{array}\right. \tag{3.19}$$

Toujours selon la même méthodologie, le Lagrangien est explicité :

$$\mathscr{L}(\alpha,\mu,\theta,\eta,\vartheta) = -\frac{1}{2}\alpha' G\alpha - \theta(\mathbf{1}'\alpha - \nu) - \mu\alpha'\mathbf{y} - \eta'(\alpha - \frac{1}{m}\mathbf{1}) + \vartheta'\alpha$$

avec μ, θ, η et ϑ des multiplicateurs de Lagrange positifs. On en déduit les relations qui permettent de résoudre le ν-SVM avec un algorithme de type SimpleSVM :

$$\begin{aligned} \alpha_w &= G_{(w,w)}^{-1}\big(\theta\mathbf{1}_w + \mu\mathbf{y}_w - \frac{1}{m}G_{(w,C)}\mathbf{1}_C\big) \\ \theta &= \frac{g_1 M + g_2 N}{OM - N^2} = \rho \\ \mu &= \frac{g_1 N + g_2 O}{OM - N^2} = b \end{aligned} \tag{3.20}$$

avec :

$$\begin{aligned} g_1 &= \left(\nu - \frac{|I_C|}{n} + \frac{1}{n}\mathbf{1}'_C G_{(C,w)} G^{-1}_{(w,w)} \mathbf{1}_w\right) \\ g_2 &= \left(\frac{1}{n}\mathbf{1}'_C \mathbf{y}_C + \frac{1}{n}\mathbf{1}'_C G_{(C,w)} G^{-1}_{(w,w)} \mathbf{y}_w\right) \\ O &= \mathbf{1}'_w G_{(w,w)} \mathbf{1}_w \\ M &= \mathbf{y}'_w G_{(w,w)} \mathbf{y}_w \\ N &= \mathbf{1}'_w G_{(w,w)} \mathbf{y}_w \end{aligned}$$

La résolution du ν-SVM pose toutefois plus de problèmes pratiques que le SimpleSVM ou le SVM à une classe. En particulier, le dénominateur $OM - N^2$ peut être nul. Ensuite, la contrainte sur la proportion de points vecteurs support contredit l'initialisation de I_w comme étant un ensemble vide. Pour résoudre ces deux problèmes, il convient d'aménager la méthode.

Tout d'abord, du point de vue des notations, l'ensemble I_C représente l'ensemble des points dont le multiplicateur de Lagrange associé est borné à $\frac{1}{n}$. La contrainte posant problème est $\alpha'\mathbf{1} \geq \nu$. Puisque la méthode de résolution est itérative, nous allons progressivement augmenter la valeur de ν au cours de la résolution jusqu'à atteindre la valeur fixée $\nu^\star$. Ainsi nous pouvons initialiser avec $\nu = \frac{1}{n}$, ce qui revient à donner aux deux premiers points sélectionnés i et j (toujours un dans chaque classe) les valeurs $\alpha_i = \alpha_j = \frac{1}{2n}$. Ensuite, au cours de la résolution, à chaque étape, on fixe $\nu = \alpha'\mathbf{1}$ tant que $\nu < \nu^\star$.

$$\begin{aligned} \alpha_w &= G^{-1}_{(w,w)}\left(\mu \mathbf{y}_w - \frac{1}{m} G_{(w,C)} \mathbf{1}_C\right) \\ \mu &= \frac{g_2}{M} = b \end{aligned} \tag{3.21}$$

Avant que le seuil $\nu^\star$ soit atteint, l'optimisation est faite sans tenir compte de la contrainte sur les sommes des α (équations 3.21). Après que le seuil est atteint, l'optimisation est faite en intégrant la contrainte (voir 3.20 page ci-contre).

Algorithme 7 : *simple-ν-SVM*

1: $(I_w, I_0, I_C) \leftarrow$ initialiser
2: $(\alpha, \mu, \theta) \leftarrow$ resoudreSystemeSansContrainte(I_w)
3: **tant que** $\nu < \nu^{star}$ **faire**
4: faire du SimpleSVM sans la contrainte de somme à $\nu^\star$
5: $\nu \leftarrow \alpha'\mathbf{1}$
6: **fin tant que**
7: **tant que** la solution n'est pas optimale **faire**
8: faire du SimpleSVM avec la contrainte de somme à $\nu^\star$
9: **fin tant que**

3.3.3 Reprise à chaud de l'algorithme

De part sa structure, la reprise à chaud de l'algorithme est immédiate. Connaissant la solution précédente, il suffit d'ajouter les nouveaux points au groupe des points inactifs I_0 et de vérifier leur classement. La violation de la contrainte de bon classement relance les itérations décrite dans l'algorithme 6 page 40.

Validation croisée La validation croisée est la méthode la plus populaire de réglage des hyper-paramètres des méthodes d'apprentissage. Cela consiste à discrétiser les espaces de chacun des hyper-paramètres à régler puis de teste successivement toutes les combinaisons possibles. L'apprentissage s'effectue sur un sous-ensemble de la base et le test sur les données restantes, que l'on appelle ensemble de validation. Il existe divers variantes, la plus commune consiste à séparer la base d'apprentissage en k ensembles et de tester k fois la sélection d'hyper-paramètres : l'apprentissage de fait sur $k-1$ sous ensembles et le test sur le $k^{\text{ième}}$ et ce successivement jusqu'à ce que chaque sous-ensemble ait servi de base de validation. La moyenne des performances en validation sert à évaluer la qualité des hyper-paramètres. L'inconvénient majeur de cette méthode est le temps de calcul. Pour p hyper-paramètres à régler pouvant chacun prendre v valeurs, le nombre d'apprentissages à effectuer est $k(v^p)$ (voir algorithme 8).

Algorithme 8 Validation croisée classique

initialiser moy_{old}, $param$
pour chaque valeur $p1$ du premier hyper-paramètre **faire**
  pour chaque valeur $p2$ du second hyper-paramètre **faire** ▷ et ainsi pour chaque hyper-paramètre
    pour chaque découpage e de la base d'apprentissage **faire**
      apprendre sur la sous-base d'apprentissage
      $perf_{(e,p1,p2)} \leftarrow$ tester sur la sous-base de validation
    fin pour
    $moy \leftarrow$ moyenne de $perf_{(:,p1,p2)}$
    si $moy > moy_{old}$ **alors**
      $moy_{old} \leftarrow moy$
      $param \leftarrow \{p1, p2\}$
    fin si
  fin pour
fin pour

La validation croisée peut bénéficier de la reprise à chaud de SimpleSVM. En effet, à chaque changement d'ensemble ou de valeurs d'hyper-paramètres, il suffit d'initialiser SimpleSVM avec les vecteurs supports de l'étape précédente comme répartition initiale. On accélère ainsi considérablement la convergence car on part d'une solution proche de celle recherchée. Pour être plus efficace, il convient de changer légèrement l'ordre des boucles car les solutions sur une même répartition des sous-ensemble sont plus proches les unes des autres que les solutions pour un même jeu d'hyper-paramètres mais sur des ensembles différents (voir algorithme 9).

Leave-one-out Le leave-one-out, cas extrême de la validation croisée, bénéficie également de la reprise à chaud. La différence avec la validation croisée réside dans le découpage de la base d'apprentissage, car on pose $k = n$. Cela revient à exclure un seul point de la base d'apprentissage et à tester sur celui-ci. Ainsi il faut faire k apprentissages par sélection d'hyper-paramètres. Cette méthode est intéressante en particulier pour des petits jeux de données.

Online SimpleSVM L'ajout successif de points un par un, comme c'est le cas dans l'apprentissage en ligne est tout aussi aisé à mettre en oeuvre. En effet, il suffit d'ajouter le point arrivant au groupe I_0 et de tester sa classification par la solution obtenue précédemment. S'il est bien classé, la méthode peut passer au point suivant. Dans le cas contraire, il est ajouté à l'ensemble des vecteurs supports I_w et la méthode est relancée jusqu'à nouvelle convergence. En pratique, pour des données stationnaires, la convergence est presque immédiate.

La mémoire est le facteur limitant de cette approche si il n'y a pas de moyen mis en oeuvre pour oublier les

Algorithme 9 Validation croisée adaptée pour tirer parti de la reprise à chaud

initialiser moy_{old}, $param$
pour chaque découpage e de la base d'apprentissage **faire**
    pour chaque valeur $p1$ du premier hyper-paramètre **faire**
        pour chaque valeur $p2$ du second hyper-paramètre **faire** ▷ et ainsi pour chaque hyper-paramètre
            apprendre sur la sous-base d'apprentissage
            $perf_{(e,p1,p2)} \leftarrow$ tester sur la sous-base de validation
        fin pour
    fin pour
fin pour
pour chaque valeur $p1$ du premier hyper-paramètre **faire** ▷ choix de la meilleure combinaison
    pour chaque valeur $p2$ du second hyper-paramètre **faire**
        $moy \leftarrow$ moyenne de $perf_{(:,p1,p2)}$
        si $moy > moy_{old}$ **alors**
            $moy_{old} \leftarrow moy$
            $param \leftarrow \{p1, p2\}$
        fin si
    fin pour
fin pour

points inutiles au fur et à mesure. Cette limitation majeure trouve une solution en suivant le fonctionnement de LASVM, qui consiste à oublier les points loin des marges.

Conclusion

Ce chapitre a présenté en détail le SimpleSVM et montré quelques unes des extensions possibles de cette méthode. Nous avons vu que la complexité de résolution est liée pour la majorité au nombre de vecteurs supports. Ainsi le caractère parcimonieux des SVM est l'atout principal des méthodes de contraintes actives.

Il existe de nombreuses extensions des SVM non présentées ici. Le SVM est une méthode de classification et possède sont pendant en régression, le SVR. Dans ce cas, les étiquettes sont des réels. L'adaptation du SimpleSVM à ce problème est aisée et existe d'ailleurs dans la boîte à outils Canu et al. [2005].

Si le problème a plus de deux classes ou alors si les sorties sont plus complexes (par exemple des graphes), on parle de sorties structurées. Ces deux derniers cas restent à mettre en œuvre mais sont des extensions envisageables. En effet la méthode décrite dans Tsochantaridis et al. [2005] est structurellement proche des contraintes actives et une perspective pour SimpleSVM est d'intégrer ces travaux.

4

Réglage des hyper-paramètres grâce au chemin de régularisation [1]

« Une fois résolu, un problème est d'une simplicité atterrante. »

Paulo Coelho

Sommaire

4.1 Le chemin de régularisation 48
4.2 Le chemin à contresens pour le ν-SVM 50
4.3 Évaluation des performances le long du chemin et arrêt anticipé 52
4.4 Résultats expérimentaux 53

DANS L'IDÉE D'APPRENTISSAGE ENDURANT, nous cherchons à fournir des méthodes capables de trouver elle-même les hyper-paramètres les plus adaptés aux données présentées. Compte tenu de l'importance de cette question pour rendre le potentiel des SVM plus accessible, beaucoup de travaux ont été menés dans cette direction. La plupart des méthodes reposent sur une évaluation externe de la performance telle que la validation croisée. D'autres méthodes utilisent une mesure incorporée à l'algorithme lui-même (pour les méthodes à noyaux, voir Argyriou et al. [2006], Micchelli and Pontil [2005]). A côté de ces méthodes empiriques pour le contrôle des paramètres, les chemins de régularisations ont été développés et largement étudiés ces dernières années Bach et al. [2005], Gunter and Zhu [2005], Hastie et al. [2004]. Les chemins de régularisation offrent une façon élégante et efficace d'accéder à toutes les solutions optimales d'un problème au regard d'un paramètre. Pour la régression, le compromis biais-variance est étudié. Dans le cas de la classification binaire qui nous intéresse ici, nous cherchons le meilleur compromis entre la performance en apprentissage et la régularité de la solution et ce avec comme objectif d'obtenir la meilleure performance en généralisation. Toutefois, connaître l'ensemble du chemin de régularisation n'est pas suffisant car il faut ensuite choisir la solution à utiliser. Pour cela nous proposons d'évaluer au fur et à mesure de l'avancement sur le chemin la performance en généralisation de la solution proposée. Cela permet d'éviter une évaluation externe ou une approximation Wahba [1999]. Pour cela nous utilisons le *leave one out*.

[1] Ce chapitre est une version remaniée du rapport technique [Loosli and Canu, 2006b]

Contrairement aux approches existantes du chemin de régularisation pour les SVM, nous proposons de commencer par la solution vide (aucun vecteur support plutôt que tous les points vecteurs supports). Puisque nous évaluons notre solution à chaque étape, nous pouvons arrêter l'apprentissage lorsque l'évaluation indique une dégradation des performances en généralisation. Ainsi il n'est pas nécessaire de calculer les solutions qui prennent beaucoup trop de vecteurs supports (et qui vont à l'encontre de l'idée de parcimonie qui est un des atouts maîtres des SVM). Éviter de calculer ces solutions permet aussi de ne pas calculer l'ensemble de la matrice de Gram et permet donc d'envisager de traiter des problèmes de plus grande dimension.

Nous allons dans un premier temps présenter les chemins de régularisation pour les SVM puis focaliser sur le ν-SVM. Nous présentons ensuite les critères d'arrêts possibles et donnons des résultats expérimentaux illustrant les intérêts de la méthode proposée.

4.1 Le chemin de régularisation

Le chemin de régularisation est constitué par l'ensemble des solutions possibles pour un problème donné en fonction d'un paramètre de compromis. Pour les SVM classiques, ce paramètre est C. A chaque valeur de C, le SVM définit les trois groupes I_0, I_w et I_C et les valeurs des coefficients α correspondants. Nous rappelons sur la figure 4.2 page ci-contre la signification de ces groupes, à savoir que :

- I_0 contient les indices des points se trouvant bien classés et du bon côté de la marge ($y_i f(x_i) + b > 1$) et par conséquent ont un coût nul (les SVM classiques utilisant le coût charnière),
- I_w contient les indices des points se trouvant sur la marge ($y_i f(x_i) + b = 1$) et se trouvant à la charnière entre un coût nul et un coût non nul,
- I_C contient les indices des points se trouvant du mauvais côté de la marge ($y_i f(x_i) + b < 1$) et ayant un coût non nul.

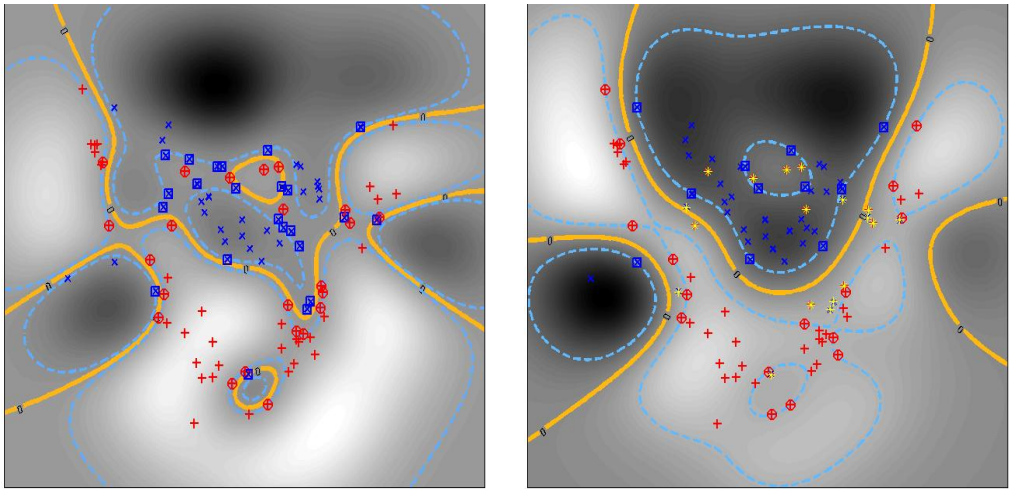


FIG. 4.1 : De l'influence du paramètre de régularisation sur la solution. A gauche, la solution sans régularisation, à droite, la solution pour 30% de vecteurs supports.

Si la solution n'est pas du tout régularisée ($C \to \infty$), aucun point ne peut se trouver dans I_C (voir figure 4.1). Si au contraire la solution est extrêmement régularisée ($C \to 0$), il se peut qu'aucun point ne trouve dans I_0. Dans ce dernier cas, on ne peut plus parler de parcimonie et utiliser les SVM n'a pas de sens. Entre ces deux extrêmes, il existe un certain nombre d'intermédiaires. Puisque les solutions aboutissant à la même répartition de points sont équivalentes et qu'il n'y a qu'un nombre fini de répartitions distinctes, le nombre de possibilités intermédiaires est

fini. Parcourir l'ensemble de ces solutions en faisant varier C produit le chemin complet de régularisation.

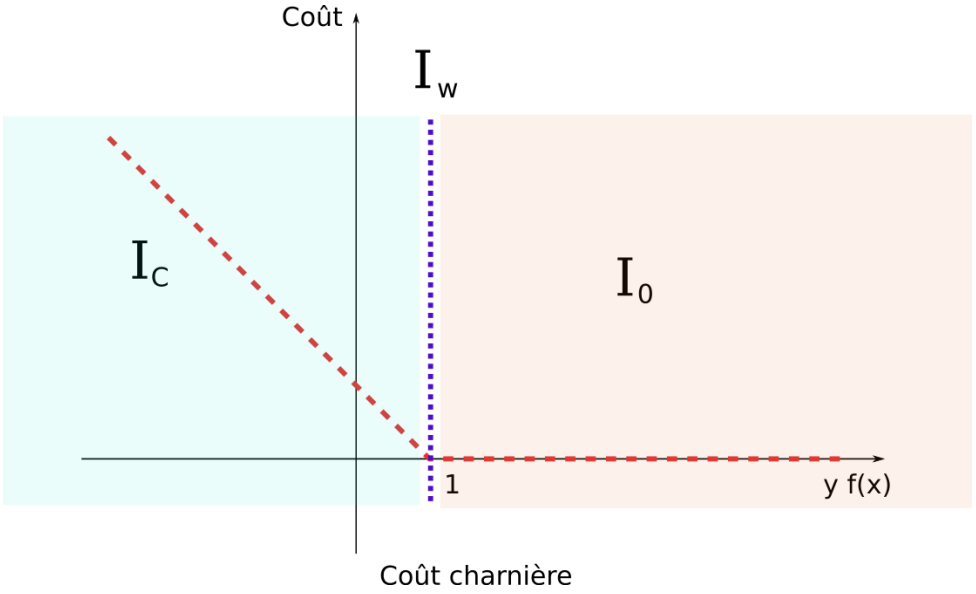


FIG. 4.2 : Fonction de coût charnière. Les groupes de points sont indiqué par rapport à leur coût. I_0 correspond à un coût nul (points bien classés), I_w correspond à un coût nul aussi mais correspond au point de coupure et I_C correspond à un coût non nul (points du mauvais côté de leur marge).

4.1.1 Un chemin linéaire par morceaux

Nous avons vu que le nombre de solutions distinctes est fini. De plus, le chemin est linéaire par morceaux et pour trouver l'ensemble de solution, il suffit de suivre une direction donnée jusqu'au prochain changement de répartition des groupes.

Le système linéaire à résoudre pour obtenir les valeurs de α_w, identique à 3.9 page 38 est affine en C. Si C varie continûment, les multiplicateurs α varient eux aussi continûment. Ainsi les variables optimales sont des fonctions de C affines par morceaux et continues.

4.1.2 Limitations

Le chemin de régularisation pour les SVM proposé par Hastie et al. [2004] choisit pour l'initialisation de mettre l'ensemble des points dans I_C. De cette façon, la première solution obtenue possède tous les points à l'intérieur ou sur les marges. En faisant grandir C, les marges rétrécissent, faisant passer les points vers I_0. Le processus est arrêté lorsqu'il ne se trouve plus un seul point dans I_C ou bien lorsqu'il n'est plus possible de réduire la marge pour les cas non-séparables. Cette approche présente quelques inconvénients auxquels nous allons essayer de répondre ici. Tout d'abord, le fait d'avoir un paramètre variant entre 0 et l'infini oblige à calculer sa première valeur et à utiliser une heuristique pour décider de sa dernière valeur dans les cas non séparables. Ensuite, en commençant par la solution la moins parcimonieuse, il est impératif de calculer l'ensemble de la matrice noyau. De ce fait, on est limité en taille de problème à résoudre.

Nous proposons dans cette partie le chemin de régularisation à contresens pour les ν-SVM. En utilisant une formulation de type ν-SVM, le paramètre à faire varier se trouve entre $\frac{1}{n}$ et 1. En commençant avec tous les points

dans I_0, on peut tirer avantage de la parcimonie. Sachant que les solutions non parcimonieuses ont peu d'intérêt, il est même possible de ne pas parcourir l'intégralité du chemin en se servant d'un critère d'arrêt. On peut alors utiliser le chemin de régularisation pour de plus grandes bases de données.

4.2 Le chemin à contresens pour le ν-SVM

Nous cherchons à connaître la solution du ν-SVM pour toutes les valeurs de ν. Comme expliqué précédemment, le chemin de régularisation est linéaire par morceaux. Cela signifie que l'ensemble des vecteurs supports est constant entre deux valeurs de ν. Ainsi il suffit d'identifier les moments où l'ensemble de vecteurs supports change pour connaître les valeurs charnières de ν. Comme pour les méthodes actives de résolution des SVM, la première étape consiste à identifier la meilleure direction à suivre et la deuxième étape sert à déterminer jusqu'où il faut suivre cette direction.

Notons $g(x_i) = y_i\big(f(x_i)+b\big) - \rho$. Alors :

$$\begin{cases} I_C : & g(x_i) < 0 & \forall i \in I_C & \alpha_i = \frac{1}{n} & \text{points bornés} \\ I_w : & g(x_i) = 0 & \forall i \in I_w & 0 < \alpha_i < \frac{1}{n} & \text{points à la marge} \\ I_0 : & g(x_i) > 0 & \forall i \in I_0 & \alpha_i = 0 & \text{points inutiles} \end{cases}$$

Chaque étape du chemin donne une solution optimale du ν-SVM pour une valeur particulière de ν. Le chemin débute par la plus petite valeur de ν continue selon les valeurs croissantes jusqu'à la valeur maximale de ν, soit 1. Puisque les solutions sont équivalentes tant que les groupes ne changent pas, nous cherchons juste à identifier les valeurs charnières pour lesquelles un point passe d'un groupe à l'autre. Il existe quatre mouvement possibles : de I_C à I_w (du mauvais côté de la marge vers à marge), de I_0 à I_w (le point devient vecteur support), de I_w à I_C (un vecteur support devient borné) et de I_w à I_0 (un vecteur support devient inutile). Ces étape sont dénotées respectivement comme *inC*, *in*0, *outC* et *out*0.

Les mouvements *inC* et *in*0 arrivent lorsque $\exists i \in I_C$ ou $i \in I_0$ tel que $g(x_i) = 0$. Les mouvements *outC* et *out*0 arrivent respectivement quand $\exists i \in I_w$ tel que $\alpha_i = \frac{1}{n}$ ou $\alpha_i = 0$.

Par conséquent il faut exprimer $g(x)$ et αen fonction des étapes t et $t+1$. Pour chaque point :

$$\begin{aligned} g^{t+1}(x_i) = &\quad g^{t+1}(x_i) - g^t(x_i) + g^t(x_i) \\ = &\quad G_{(i,:)}\alpha^{t+1} + b^{t+1}y_i - \rho^{t+1} - G_{(x_i,:)}\alpha^t - b^t y_i + \rho^t + g^t(x_i) \\ = &\quad G_{(i,:)}\delta_\alpha + \delta_b y_i - \delta_\rho + g^t(x_i) \end{aligned} \tag{4.1}$$

avec $\delta_\alpha = \alpha^{t+1} - \alpha^t$, $\delta_b = b^{t+1} - b^t$ et $\delta_\rho = \rho^{t+1} - \rho^t$. $G(i,:)$ désigne la $i^{\text{ème}}$ ligne de la matrice. De l'équation 4.1 et en fonction de l'événement concerné, il est possible de trouver la valeur de ν à choisir pour la suite. Le tableau 4.1 résume les événement et la mécanique de l'algorithme.

4.2.1 Points de I_w et détection de *outC* et *out*0

Les événements *outC* et *out*0 sont détectés en utilisant leur valeurs de α. En effet, une condition pour rester dans I_w est de conserver la propriété $0 < \alpha < \frac{1}{n}$. Rechercher ν^{t+1} pour lequel cette condition est violée pour chacun des points requière l'écriture de α_i en fonction de ν^{t+1}

Remarquons que dans I_w, $g^t(x) = 0$ et $g^{t+1}(x) = 0$. L'équation 4.1 assortie des contraintes ($\alpha'\mathbf{1} \geq \nu$, donc $\delta'_\alpha \mathbf{1} \geq \nu^{t+1} - \nu^t$ and $\alpha'\mathbf{y} = 0$, par conséquent $\delta'_\alpha \mathbf{y} = 0$) conduit à un système d'équations linéaires $A\delta = (\nu^{t+1} -$

Étape	*in*0	*out*0	*outC*	*inC*
t	$i \in I_0$ $g^t(x_i) > 0$ $\alpha_i = 0$	$i \in I_w$ $\star\mathbf{g^t(x_i) = 0}$ $0 < \alpha_i < \frac{1}{n}$	$i \in I_w$ $\star\mathbf{g^t(x_i) = 0}$ $0 < \alpha_i < \frac{1}{n}$	$i \in I_C$ $g^t(x_i) < 0$ $\alpha_i = \frac{1}{n}$
$t+1$	$i \in I_w$ $\star\mathbf{g^{t+1}(x_i) = 0}$ $0 < \alpha_i < \frac{1}{n}$	$i \in I_0$ $\star\mathbf{g^{t+1}(x_i) \geq 0}$ $\star\mathbf{\alpha_i = 0}$	$i \in I_C$ $\star\mathbf{g^{t+1}(x_i) \leq 0}$ $\star\mathbf{\alpha_i = \frac{1}{n}}$	$i \in I_w$ $\star\mathbf{g^{t+1}(x_i) = 0}$ $0 < \alpha_i < \frac{1}{n}$

TAB. 4.1 : Résumé des événements. Chaque colonne concerne un événement particulier. En bleu étoilé sont indiquées les propriétés utilisées pour le calcul du ν^{t+1} correspondant

$\nu)\mathbf{c}$, avec

$$A = \begin{bmatrix} G & \mathbf{y} & -\mathbf{1} \\ \mathbf{y}' & 0 & 0 \\ \mathbf{1}' & 0 & 0 \end{bmatrix} \quad \delta = \begin{bmatrix} \delta_\alpha \\ \delta_b \\ \delta_\rho \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} \mathbf{0} \\ 0 \\ 1 \end{bmatrix}$$

Cela amène à $\delta = (\nu^{t+1} - \nu^t)A^{-1}\mathbf{c}$ donc pour les points de I_w :

$$\begin{cases} \alpha^{t+1} = \alpha^t + (\nu^t - \nu^{t+1})\mathbf{d}_\alpha \\ b^{t+1} = b^t + (\nu^t - \nu^{t+1})\mathbf{d}_b \\ \rho^{t+1} = \rho^t + (\nu^t - \nu^{t+1})\mathbf{d}_\rho \end{cases}$$

avec $\mathbf{d}$ désignant le vecteur $A^{-1}\mathbf{c}$.

Comme mentionné précédemment, les groupes changent dès que l'un des α_i rencontre une borne, c'est-à-dire lorsque $\alpha_i = 0$ ou $\alpha_i = \frac{1}{n}$ pour $i \in I_w$. Cela donne deux conditions de changement :

$$\begin{cases} \nu^{t+1}_{out0} = \dfrac{-\alpha^t_i}{\mathbf{d}_i} + \nu^t \\ \nu^{t+1}_{outC} = \dfrac{\frac{1}{n} - \alpha^t_i}{\mathbf{d}_i} + \nu^t \end{cases}$$

4.2.2 Points de I_C et I_0 et détection de *inC* et *in*0

Les événements *inC* et *in*0 sont détectés par leur valeur de $g(x_i)$. En effet, une condition pour rester dans I_0 est de conserver la propriété $g(x_i) > 0$ et de même, conserver $g(x_i) < 0$ permet de rester dans I_C. Rechercher ν^{t+1} pour lequel une de ces condition est violée pour chaque point nécessite d'écrire $g^{t+1}(x_i)$ en fonction de ν^{t+1}. Pour un point entrant dans I_w, la particularité utilisée est que $g^{t+1}(x_i)$ devient nul.

En définissant $h(x) = G(i,:)\mathbf{d}_\alpha + \mathbf{d}_b - \mathbf{d}_\rho$:

$$\begin{aligned} g^{t+1}(x_i) = &\quad G(i,:)\delta_\alpha + \delta_b y_i - \delta_\rho + g^t(x_i) \\ = &\quad (\nu^{t+1} - \nu^t)\big(G(i,:)\mathbf{d}_\alpha + \mathbf{d}_b - \mathbf{d}_\rho\big) + g^t(x_i) \\ = &\quad (\nu^{t+1} - \nu^t)h^t(x_i) + g^t(x_i) = 0 \end{aligned}$$

et ainsi

$$\begin{aligned} \nu^{t+1}_{inC} = &\quad \frac{-g^t(x_i)}{h^t(x_i)} + \nu^t \quad i \in I_C \\ \nu^{t+1}_{in0} = &\quad \frac{-g^t(x_i)}{h^t(x_i)} + \nu^t \quad i \in I_0 \end{aligned}$$

Remarquons que cela peut arriver que plusieurs points atteignent I_w au même moment. Cette remarque est surtout utile pour des aspects d'implémentation. En effet, manquer un point dévie définitivement du chemin de régularisation car il ne sera plus jamais sélectionné par la suite.

4.2.3 Algorithme du chemin de régularisation

A chaque étape, le plus petit $\nu^{t+1} > \nu^t$ parmi $\left\{\nu_{inC}^{t+1}, \nu_{in0}^{t+1}, \nu_{outC}^{t+1}, \nu_{out0}^{t+1}\right\}$ est recherché. Les α^{t+1} sont mis à jour en fonction du ν^{t+1} choisit et ensuite les groupes sont redéfinis. Ce processus s'arrête lorsque $\nu = 1$.

L'étape de résolution de système linéaire est similaire à celle du Simple-ν-SVM. Ainsi nous avons intégré les deux méthodes en une seule. De cette façon, lorsque le chemin est quitté (pour des raisons de stabilité numérique par exemple), il suffit de quelques étapes de ν-SVM pour revenir sur le chemin. A chaque fois que le système linéaire est mis à jour, on vérifie si la solution est toujours optimale. Cet ajout ne coûte rien de plus car il utilise les calculs nécessaires à la recherche de la prochaine valeur de ν. Si la solution est optimale, on procède à la recherche de ν. Dans le cas contraire, on revient à une solution optimale sans toucher à ν (voir algorithme 4.2.3).

Initialisation L'initialisation est faite pour $\nu = \frac{1}{n}$, ce qui signifie qu'aucune erreur est tolérée dans l'apprentissage. Le Simple-ν-SVM est utilisé à cet effet. La première solution est la plus parcimonieuse des solutions sans erreur d'apprentissage sur l'ensemble du chemin (et potentiellement la plus sur-apprise). Si le problème est réellement séparable, cette première solution est la meilleure. Cette initialisation peut ne pas être possible même avec un noyau gaussien, comme souligné dans Hastie et al. [2004]. Cela arrive lorsque la matrice noyaux n'est pas de rang plein. Dans ce cas, il est facile de détecter ce problème (l'erreur en apprentissage est non nulle) et il est possible d'initialiser le chemin avec une plus grande valeur de ν. Une autre solution consiste à réduire la largeur de bande du noyau.

Algorithme 10 Chemin de régularisation du ν-SVM

```
initialisations, ν = 1/n
tant que ν < 1 faire
    résoudre le système linéaire                          ▷ Calcul des α_w
    si solution non admissible alors
        projeter la solution dans le domaine admissible   ▷ étape Simple-ν-SVM
    sinon si solution non optimale alors
        sélectionner le prochain point                    ▷ étape Simple-ν-SVM
    sinon
        calculer le prochain ν                            ▷ étape du chemin
    fin si
fin tant que
```

4.3 Évaluation des performances le long du chemin et arrêt anticipé

Sachant parcourir tout le chemin de régularisation et sachant que le point de départ est la solution la plus sur-apprise, l'objectif est de parcourir le chemin vers une solution régulière mais encore parcimonieuse et de s'y arrêter. Pour cela il faut évaluer l'erreur en généralisation à chaque étape. Nous étudions donc s'il est possible de trouver un critère d'arrêt sur le chemin.

Parmi toutes les estimations de l'erreur en généralisation possibles, la méthode du *leave-one-out* est celle qui est recommandée par les praticiens. Cependant, le principal inconvénient de cette approche est le temps de calcul. Des solutions ont été proposées pour répondre à ce problème.Wahba [1999] propose d'utiliser une approximation appelée GCV (*Generalized Cross Validation*). D'autres comme Lee and Lin [2000] proposent de tirer avantage des implementations efficaces des SVM avec reprise à chaud pour dériver des procédures acceptables. Ici nous suivons cette approche et montrons comment intégrer cet estimateur au sein du chemin de régularisation de façon à gagner en temps de calcul par rapport à une procédure externe.

L'erreur en *leave-one-out* est définie comme la moyenne des erreur commises sur le point enlevé. On peut aussi calculer un estimateur à partir de la variance des solutions :

$$LOO1_{erreur} = \frac{1}{n}\sum_i 1 - sign(\hat{y}_i y_i) \qquad LOO2_{erreur} = \frac{1}{n}\sum_i max(0, \rho - \hat{y}_i y_i)$$

Ce second estimateur est utile pour détecter le sur-apprentissage.

L'erreur en *leave-one-out* est calculée à chaque étape du chemin de régularisation. Puisqu'aucun point de I_0 ne participe à la solution, ils ont nécessairement une erreur nulle s'ils sont enlevés de l'ensemble d'apprentissage. Ainsi il n'est pas utile de faire ce calcul et nous pouvons nous contenter de calculer les erreurs pour chaque point ℓ de I_w et I_C. La solution $S_{-\ell}$ est proche de la solution courante S trouvée sur le chemin donc nous obtenons $S_{-\ell}$ à partir de S avec la reprise à chaud de Simple-ν-SVM. Le coût de chaque estimation LOO est $\left(|I_w| + |I_C|\right)$ étapes de mise à jour. Puisque Simple-ν-SVM fonctionne de manière très similaire au chemin de régularisation et utilise les même groupes, la convergence entre deux solutions proche est rapide.

Si l'erreur en généralisation est significativement meilleure pour certains paramètres ν, cela doit se voir au travers des estimations LOO. L'estimateur LOO est donc monitoré au cours du chemin pour détecter une augmentation durable et arrêter l'apprentissage. Ensuite il est possible de revenir à la solution S_ν qui a donné le minimum d'erreur LOO. De cette façon on évite le calcul d'une partie du chemin ainsi que l'évaluation LOO de cette partie. Par ailleurs, calculer les solutions non parcimonieuses n'est pas *a priori* utile.

4.4 Résultats expérimentaux

4.4.1 Illustration des chemins

Les chemins de régularisation peuvent être représentés à l'aide des valeurs successives des multiplicateurs α au cours de l'apprentissage. Ces multiplicateurs suivent un chemin linéaire par morceaux. La figure 4.3 page suivante illustre ce comportement pour le chemin de régularisation du ν-SVM. De manière à rendre cette figure plus claire, les multiplicateurs sont multipliés par l'étiquette du point correspondant.

4.4.2 Efficacité du chemin de régularisation du ν-SVM

Le chemin de régularisation du ν-SVM obtient, sur les données « mixture », les même résultats que le chemin pour le C-SVM de Hastie et al. [2004], en termes de précision, nombre de changement de groupes et vitesse d'exécution. La seule différence est le fait de suivre le chemin dans le sens opposé. Cela présente des avantages pratiques lors de l'introduction du critère d'arrêt LOO.

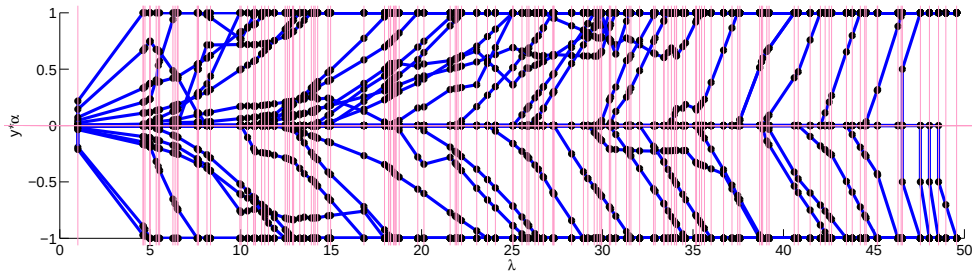


FIG. 4.3 : Évolution des $y_i \alpha_i$ le long du chemin pour le problème « pomme-banane » (annexe 7.3.5 page 111)

4.4.3 Sur le critère d'arrêt

Pour illustrer les estimations LOO et leur rôle en temps que critère d'arrêt, nous avons calculé l'ensemble des chemins et évaluations LOO pour les deux jeux de données artificielles. Les figures 4.4 et 4.5 représentent respectivement les résultats pour les jeux « pommes-banane » et « mixture » avec noyau gaussien. Chaque estimateur LOO fournit une information utile. Le premier, basé sur le comptage des erreurs donne une bonne estimation de l'erreur de généralisation tandis que l'autre, basée sur la variance des sorties des SVM représente la variabilité de la solution (une grande variance est signe de sur-apprentissage) et nous cherchons une solution à faible variance.

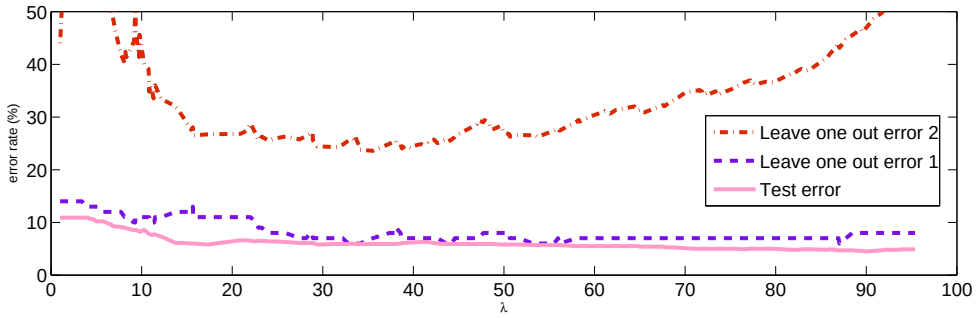


FIG. 4.4 : Illustration sur le jeu « pomme-banane » (annexe 7.3.5 page 111) de l'erreur LOO en fonction de ν, donné avec l'erreur de test

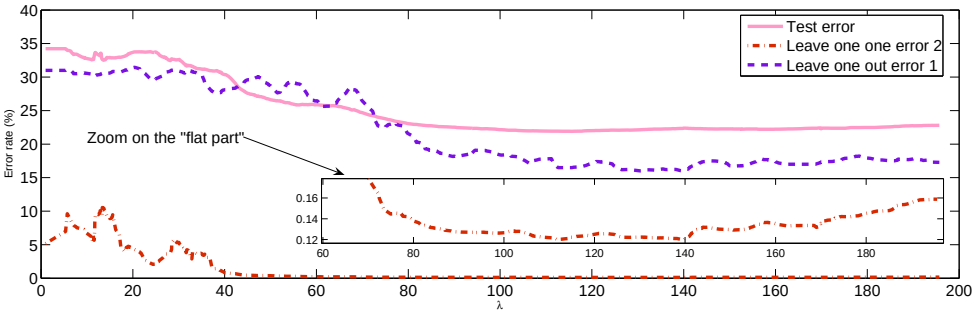


FIG. 4.5 : Illustration sur le jeu « mixture » (annexe 7.3.5 page 111) de l'erreur LOO en fonction de ν, donné avec l'erreur de test

D'un point de vue pratique, déterminer le moment d'arrêt requiert l'utilisation d'heuristiques et de courbes lissées des estimations LOO. L'heuristique utilisée consiste à choisir de s'arrêter quand l'estimation lissée LOO2 ne décroît plus pendant une période qui correspond à l'accroissement de ν de 10. Ensuite nous revenons en arrière pour sélectionner le meilleur ν, obtenu grâce à l'estimation LOO1.

Conclusion

L'approche présentée dans ce chapitre permet de s'affranchir un peu plus de l'intervention humaine ou exterieure dans l'utilisation des SVM. Le chemin de régularisation permet l'auto-calibration du compromis biais-variance grâce à la mise en place d'un critère d'arrêt beaucoup moins coûteux qu'une procédure externe.

Contrairement à Hastie et al. [2004], nous ne partons pas de la solution prenant tous les exemples comme vecteurs supports. Au contraire, nous partons de la solution la plus parcimonieuse possible et le chemin est parcouru en direction de la solution la plus dense. Nous arrêtons le parcours avant d'atteindre la solution dense. Pour déterminer le point d'arrêt, nous évaluons la qualité de la solution courante à chaque point d'arrêt à l'aide de la méthode du *leave-one-out*. Ainsi il est possible de traiter de plus grandes bases de données avec un chemin de régularisation.

Troisième partie

Applications et mise en œuvre

5

En pratique : boîte à outils SimpleSVM [1]

« La théorie, c'est quand on sait tout et que rien ne fonctionne.
La pratique, c'est quand tout fonctionne et que personne ne sait pourquoi.
Ici, nous avons réuni théorie et pratique :
Rien ne fonctionne... et personne ne sait pourquoi ! »

Albert Einstein

Sommaire

5.1 Une boite à outils en Matlab 59
5.2 Étude des critères d'arrêt et comparaison des méthodes de résolution 64

CE CHAPITRE est consacré à l'implémentation de SimpleSVM et à l'étude des corportements induits par les aspects machine. L'algorithme de contraintes actives pour les SVM existait dès 1998 dans la boîte à outils [Canu et al., 2005] sous le nom *monqp*. Après étude de la répartition du temps de calcul il est apparu qu'un certain nombre d'améliorations étaient possibles. De là est née la boîte à outils SimpleSVM [Loosli, 2004], ainsi nommée en réference aux travaux de Vishwanathan et al. [2003] sur le même algorithme de contraintes actives. Dans les parties qui suivent nous décrivons les aspects les plus techniques de cet algorithme ainsi que l'implémentation faite en Matlab. Ensuite nous expliquons comment dériver ses programmes quadratiques et donnons des exemples existant tels que les SVM à une classe ou encore les SVM invariants.

5.1 Une boite à outils en Matlab

La phase de mise en œuvre d'un algorithme est l'ultime test permettant d'éprouver ses qualités. Elle peut révéler une certaine instabilité ou un autre défaut fatal, ayant échapé à l'analyse théorique, souvent focalisée sur les aspects positifs de la méthode. Nous allons présenter ici les différents aspects liés à la mise en œuvre d'un algorithme permettant de calculer la solution du problème des SVM. Cet algorithme du type « contraintes actives » ou « gradient projeté », publié sous le nom de *SimpleSVM* [Vishwanathan et al., 2003], a passé l'épreuve avec

[1] Les travaux contenus dans ce chapitre ont donnés lieu aux publications [Loosli, 2004, Loosli and Canu, 2006a]

succès. Il s'est révelé être non seulement très efficace en terme de temps de calcul et de précision mais aussi fertile en extensions potentielles. L'ensemble des programmes dont nous allons parler est disponible en ligne[2].

5.1.1 Initialisation

Le principe général de l'algorithme est par défaut de commencer par très peu de points. Ainsi, les premières itérations sont très rapides. La stratégie la plus simple pour initialiser le processus est le tirage aléatoire d'un point par classe. Elle est utilisée par défaut. Il existe d'autres critères pour choisir un point par classe. On peut par exemple les choisir de sorte que la distance qui les sépare soit la plus courte possible. On est ainsi sûr de commencer avec des points qui seront vecteurs supports [Roobaert, 2000].

On peut aussi procéder en projetant la solution calculée sans contraintes sur un sous-ensemble des points, à l'intérieur du domaine admissible, constituant ainsi trois groupes de points (saturés à C, actifs et inactifs). On met dans le groupe des saturés tous les points tels que $\alpha \geq C$, dans le groupe des inactifs tous les points tels que $\alpha \leq 0$ et les derniers comme solution candidate dans le groupe des supports.

5.1.2 Résolution du système linéaire

Le cœur de l'algorithme et aussi la partie la plus gourmande en temps de calcul est la résolution du système de la forme :

$$\begin{pmatrix} G & \mathbf{y} \\ \mathbf{y}' & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \mu \end{pmatrix} = \begin{pmatrix} \mathbf{e} \\ f \end{pmatrix}. \tag{5.1}$$

De façon à pouvoir garder le bénéfice de la positivité de la matrice G, le système est reformulé en découplant les variables. En définissant $M = G^{-1}\mathbf{y}$ et $N = G^{-1}\mathbf{e}$, α et μ sont donnés par :

$$\begin{cases} \mu = \dfrac{f - \mathbf{y}'}{M\mathbf{y}'N} \\ \alpha = N - \mu M \end{cases} \tag{5.2}$$

Deux stratégies différentes peuvent être utilisées pour résoudre le système linéaire de type $Gx = b$. Puisque nous savons la matrice G définie positive, la factorisation de Cholesky est la plus efficace. Mais dans le cas où la matrice G est singulière il est préférable d'utiliser la factorisation QR.

En pratique pour résoudre notre problème, voici comment utiliser la décomposition de Choleski :

```
U = chol(G);
M = U\Ut\y;           % Ut désignant la transposée de U
N = U\Ut\e;
mu=(yt*M)/(yt*N);
alpha=N-mu*M;
```

La factorisation QR s'écrit $G = QR$ avec R une matrice triangulaire supérieure et Q une matrice orthogonale ($Q'Q = I$). Nous utilisons cette décomposition de manière classique. Elle s'écrit en Matlab :

[2] http://asi.insa-rouen.fr/~gloosli

```
[Q,R] = qr(G);
M=R\Qt*y;              % Qt désignant la transposée de Q
N=R\Qt*e;
```

5.1.3 Résolution Dynamique

Sachant que nous avons besoin de refaire ce calcul à chaque itération de l'algorithme avec seulement un point de différence par rapport à l'itération précédente, il est intéressant de regarder les méthodes incrémentales dont la complexité est $\mathcal{O}(|I_w|^2)$ plutôt que les méthodes directes en $\mathcal{O}(|I_w|^3)$. Les deux algorithmes de factorisation (Choleski et QR) peuvent être traités de la sorte. Il est aussi possible de calculer incrémentalement G^{-1} l'inverse de la matrice G, en utilisant la formule de Sherman Morrison.

5.1.4 Ajout d'un point dans la solution

Plusieurs stratégies peuvent être adoptées lors de la phase d'ajout d'un point dans le groupe des supports. La première, la plus proche de la formulation mathématique, consiste à prendre tous les points des groupes saturés et inactifs et vérifier que les contraintes sont bien satisfaites pour tous. Si l'on travaille sur une grande base de données, on se réalisera qu'aussi bien en mémoire qu'en calculs, cette opération demande beaucoup de ressources. Notre idée est de segmenter cet ensemble de points et de chercher un point à ajouter sur des sous-groupes de p points. S'il n'y a aucun point à ajouter dans le premier groupe, on regarde dans le suivant jusqu'à trouver un point. Si après le passage en revue de tous les points on n'a rien trouvé, alors nous avons la solution finale. Au contraire dès que l'on trouve un point, on le rajoute sans regarder plus loin et on revient à la première étape avec notre nouveau jeu d'actifs. Plus le pas de segmentation p est petit, plus le calcul de vérification de l'optimalité est rapide. En revanche plus le pas est petit, plus on choisit de points qui seront rejetés par la suite car on ne prend pas à chaque itération le pire de tous, mais le pire de quelques uns. On se retrouve donc à devoir choisir un compromis entre le nombre d'itérations et la taille du calcul à chaque itération. En pratique nous avons constaté qu'un pas de l'ordre d'une centaine de points est en général un bon compromis. Il est à noter ici que si l'on envisage de traiter des problèmes qui requièrent un traitement particulier à chaque point (par exemple si l'on veut intégrer des invariances, on regardera le point et toutes ses variations pour savoir si on l'ajoute) alors le pas devra être $p = 1$ et cela ralentira d'autant la résolution.

5.1.5 Critère d'arrêt

Nous avons vu que l'algorithme converge (3.2.3), mais pour des raisons de précision machine nous définissons une borne d'erreur ε. Nous admettons par là que l'algorithme a convergé lorsque chacun des points contribue pour moins de ε au saut de dualité [Schölkopf and Smola, 2002]. En pratique, la valeur de ε influe peu sur le temps de calcul contrairement aux méthodes de type SMO, pour lesquelles le nombre d'itérations croît avec la précision demandée.

5.1.6 Cache

Un point clef de l'implémentation efficace des SVM concerne le calcul du noyau. La méthode la plus rapide en temps de calcul est de garder en mémoire tous les éléments du noyau qui sont calculés une fois afin de gagner du

temps pour une réutilisation ultérieure. La méthode la plus économique en mémoire est de calculer des éléments du noyau à chaque fois que l'on en a besoin et de ne rien stocker. La bonne pratique se situe entre ces deux extrêmes et consiste en un compromis entre mémoire et vitesse de calcul. Il existe des techniques très avancées de gestion de cache [Joachims, 1999]. Le cache contient toutes les valeurs du noyau utilisées les plus fréquemment dans la limite de la place mémoire disponible. Notre boîte à outils contient une politique de cache simple qui gère la partie du noyau qui correspond aux vecteurs supports. Cette approche est sans conteste plus efficace que de pré-calculer le noyau complet mais souffre encore de quelques défauts (en particulier, on ne peut pas fixer de taille maximale) qui seront traités dans les prochaines versions.

5.1.7 Fonctionnement général

Voici un exemple d'utilisation de la boite à outils, pour la résolution d'un problème binaire artificiel, le damier (annexe 7.3.5 page 111).

```
global svModel
[x,y,xt,yt]=dataset('Checkers',200,50,0.5); % genère des données
donnees = data(x,y,xt,yt);                  % créé la structure des données
noyau = kernel('rbf',.9);                   % créé la structure du noyau
parametres = param(500,50,'binary','chol'); % créé la structure des paramètres
trainSVM(donnees, noyau, parametres);       % entraine le SVM
prediction = testSVM;                       % test le modèle sur les données de test
                                            % (et renvoie f(x)+b pour chaque point)
```

5.1.8 Description des structures et des principales fonctions

La structure `svModel` contient les quatres structures précédement citées :

- les données,
- le noyau,
- les paramètres,
- le modèle

Les données Cette structure, créée par l'appel de la fonction `data` , contient la base d'apprentissage avec les étiquettes ainsi que la base de test. Elle peut aussi contenir, dans le cas du traitement des invariances, les exemples virtuels (voir chapitre 6 page 71).

```
mydata = data(trainMatrix, trainLabels, testMatrix, testLabels) ;
```

`mydata` obtenu :

```
mydata =

      trainvec: [2x96 double]
      trainlab: [96x1 double]
       testvec: [2x48 double]
```

```
       testlab: [48x1 double]
    trainvec_t: []
    trainlab_t: []
     testvec_t: []
     testlab_t: []
```

Les champs vides servent à garder en mémoire les données d'origine lorsqu'elles sont transformées au cours de l'apprentissage.

Le noyau Le noyau est créé en faisant appel à la fonction `kernel` :

```
mykernel = kernel('rbf', 0.9);
```

`mykernel` obtenu :

```
mykernel =

     name: 'rbf'
    sigma: 0.9000
```

Paramètres La structure créée par l'appel à `param` contient toutes les informations utiles à l'algorithme :

```
parametres = param(C,step,multiclass,method,gap,type,transfo);
```

```
myparam =
                  C: 500              % Paramètre de régularisation
               step: 50               % Nombre de points à chaque étape
         multiclass: ''               % '1vs1', '1vsall', '1class',...
             method: 'chol'           % Résolution du système linéaire
                gap: 1.0000e-005      % KKT tolérance
               type: 'none'           % variations du simpleSVM: 'invariant'
    transformations: []               % Liste des transformations (invariances)
```

Modèle La structure créée par l'algorithme contient tous les résultats pour évaluer de nouvelles données mais aussi pour reprendre à chaud.

```
svModel.model =

     iclass: 1                        % nombre de classifier stockés
      alpha: {[117x1 double]}         % multiplicateurs de Lagrange
    indices: {[117x1 double]}         % liste des indices des vecteurs supports
         mu: {[-1.0831]}              % biais
         G1: {[500x500 double]}       % noyau stocké
         G2: {[500x500 double]}       % décomposition du noyau
        nbC: 53                       % nombre de vecteurs support bornés à C
```

dataset Cette fonction permet de générer des données artificielles.

5.2 Étude des critères d'arrêt et comparaison des méthodes de résolution

Nous nous intéressons maintenant à l'étude des critères d'arrêt des différentes méthodes de résolution des SVM hors ligne. Nous comparons les effets du critère d'arrêt pour C-SVM (SVM classique noté ainsi pour le distinguer des autres versions comparées ici), ν-SVM et CVM. Le critère d'arrêt est dépendant de l'implémentation choisie. Les résultats reportés ici sont obtenus pour notre implémentation de SimpleSVM ainsi qu'avec libSVM pour SMO.

5.2.1 Notations et définitions

G est le noyau tel que $G_{ij} = y_i y_j k(x_i, x_j)$. $\tilde{G}$ est le noyau modifié tel que $\tilde{G}_{ij} = y_i y_j k(x_i, x_j) + y_i y_j + \frac{\delta_{ij}}{C}$ ($\delta_{ij} = 1$ si $i = j$ et 0 sinon). $\tilde{G}$ est utilisé pour la reformulation du SVM en MEB (voir section 2.3.1).

Toutes les expériences décrites ici peuvent être reproduites avec les codes et données publiées :

- **CVM [Tsang et al., 2005]** Version 1.1 (en C) de `http://www.cs.ust.hk/~ivor/cvm.html` pour Linux.
- **LibSVM [Chang and Lin, 2001a]** Version 2.7 livrée avec le code de CVM et 2.81-1 [Rong-En Fan, 2005].
- **SimpleSVM [Loosli, 2004, Vishwanathan et al., 2003]** Version 2.3 de `http://asi.insa-rouen.fr/~gloosli/coreVSsimple.html`.
- **Données** Les données sont disponibles sur `http://asi.insa-rouen.fr/~gloosli/coreVSsimple.html` en formats libSVM et SimpleSVM.

5.2.2 C-SVM (SimpleSVM et SMO)

Le critère d'arrêt classique utilisé pour le C-SVM est :

$$\min(G\alpha_1 - \mathbf{1}) \geq -\varepsilon_1 \tag{5.3}$$

Les deux implémentations choisies, SimpleSVM et libSVM utilisent un critère similaire à celui-ci, au biais près. Cela correspond à la contrainte de bonne classification. Ici l'ordre de grandeur de α, influencé par C, modifie légèrement la sensibilité du critère. Pour SMO, un critère très strict implique un grand nombre d'itérations pour converger et ralentit l'apprentissage. C'est la raison pour laquelle il est communément admit que SMO ne doit pas être utilisé avec de grandes valeurs de C (par ailleurs cette idée est souvent étendue aux SVM en eux-mêmes, ce qui est faux). La vitesse de convergence du SimpleSVM est quant à elle liée au nombre de vecteurs supports et n'est pas pénalisée par un critère plus strict (il n'y a pas en effet de recherche de direction par gradient du premier ordre). Pour un problème séparable, un grand C donne une solution plus parcimonieuse et donc plus rapide.

5.2.3 ν-SVM

Pour le ν-SVM le critère d'arrêt [Chen et al., 2005] est :

$$\min(G\alpha_2 - \rho_2 \mathbf{1}) \geq -\varepsilon_2 \tag{5.4}$$

avec ρ_2 correspondant aux marges [Chang and Lin, 2001b]. Pour pouvoir comparer ce critère au précédent, il est possible de mettre en relation les deux formulations et d'établir les liens qui permettent d'obtenir la même fonction de décision. Pour cela, prenons un problème fixé, un ε_2 et la solution optimale du ν-SVM résumée par α_2 et ρ_2. Pour qu'un C-SVM donne la même frontière, on utilise les relations $C = 1/\rho_2$ [Schölkopf and Smola, 2002, p209, prop. 7.6] et $\varepsilon_1 = \varepsilon_2/\rho_2$. La solution α_1 obtenue par le CVM est aussi liée à α_2 puisque $0 \leq \alpha_2 \leq 1$ et $0 \leq \alpha_1 \leq C$. Ainsi, nous avons $\alpha_2 = \alpha_1/C$. Avec ces relations, le critère d'arrêt de C-SVM peut être exprimé avec les termes du ν-SVM :

$$\varepsilon_1 = \varepsilon_2 C = \frac{\varepsilon_2}{\rho_2}$$

Avec un petit ε_2, une petite marge trouvée par ν-SVM conduit à un grand C (par exemple $1/\varepsilon_2$). Le C-SVM équivalent devrait être lancé avec $\varepsilon_1 = 1$ ce qui est très lâche. A cause de cet effet d'échelle, toute comparaison doit être prudente.

5.2.4 CVM

Dans cette partie nous montrons que *CVM* et ν-SVM ont un critère d'arrêt similaire d'un point de vue de mise à l'échelle. En dépit de cela, nous allons voir que la comparaison directe entre CVM et les autres méthodes n'est toutefois pas aisée. Par ailleurs, nous mettons en évidence l'impact de l'ordre de grandeur de C sur le comportement du CVM.

Rappelons ici que CVM utilise deux boucles imbriquées (voir l'algorithme 4 page 27). La boucle externe sert à la sélection du *coreset* tandis que la boucle interne calcule les multiplicateurs de Lagrange.

Considérons tout d'abord la boucle interne, appliquée aux points du *coreset*. Les multiplicateurs α_3 sont calculés en utilisant SMO. Toutefois, en suivant Tsang et al. [2005, équation 5], le problème résolu est similaire à un ν-SVM. Ainsi le critère d'arrêt est similaire à celui donné précédemment. Il existe cependant une différence qui réside dans le noyau utilisé : CVM utilise un noyau dans lequel C intervient explicitement.

Considérons ensuite la boucle externe, c'est-à-dire les points qui ne sont pas utilisés par le SMO. Le critère d'arrêt est donné par Tsang et al. [2005, eq. 25] :

$$\min\big((\tilde{G}\alpha_3)_{I_{ncs}} - \rho_3 \mathbf{1}\big) \geq -\varepsilon_3 \tilde{\mathrm{K}} \tag{5.5}$$

Si la solution α_3 est obtenue avec le CVM pour un ε_3 donné, puisque $\mathbf{1}'\alpha_3 = 1$, la solution équivalente α_1 donnée par un C-SVM est reliée de la manière suivante : $\alpha_3 = \alpha_1/(\mathbf{1}'\alpha_1)$. De plus, en posant $\rho_3 = 1/(\mathbf{1}'\alpha_1)$ et en utilisant ces relations ainsi que l'expression développée du noyau on obtient :

$$\min\Big(\frac{\big((G + \mathbf{y}\mathbf{y}' + \frac{1}{C}Id)\alpha_1\big)_{\mathscr{R}} - \mathbf{1}}{\mathbf{1}'\alpha_1}\Big) \geq -\varepsilon_3(\mathrm{K} + 1 + \frac{1}{C}) \tag{5.6}$$

Ici, de la même façon, il est possible d'exprimer la condition d'arrêt du C-SVM avec les termes du CVM :

$$\varepsilon_1 = \frac{\varepsilon_3(\mathrm{K} + 1 + \frac{1}{C}) + \big((\mathbf{y}\mathbf{y}' + \frac{1}{C}Id)\alpha_3\big)_\ell}{\rho_3} \tag{5.7}$$

où ℓ est l'indice du point minimisant la partie gauche de (5.6). On observe un effet d'échelle similaire à celui du ν-SVM.

En laissant de côté le souci de comparaison équitable entre les méthodes, l'analyse précédente met en avant l'influence non négligeable de C sur le comportement de CVM en fonction des paramètres fournis. En revenant à l'expression 5.5 page précédente :

- Le noyau modifié cache un effet régularisant. En effet, le terme $1/C$ est ajouté à la diagonale. Si pour une grande valeur de C ce terme est négligeable, ce n'est pas le cas pour des petites valeurs qui rendent le noyau très bien conditionné . De cette façon on peut s'attendre à des résultats plus précis pour des petites valeurs de C,
- à cause que sa forme analogue à ν-SVM, le critère devient plus strict pour de petites valeurs de C. Cela a deux effets. Le premier, dû à SMO, est l'augmentation du nombre d'itérations avant convergence et le deuxième est un effet secondaire de l'augmentation du nombre d'itérations : plus de points sont vérifiés par la boucle externe lors du tirage au hasard des 59 points.

Globalement, une petite valeur pour C doit entraîner une résolution plus lente que SMO et des résultats précis. Une grande valeur pour C au contraire devrait donner des résultats moins précis. Par ailleurs, la taille de l'ensemble d'apprentissage joue aussi sur l'effet d'échelle. Par conséquent, les effets de C sont amplifiés pour des grandes bases d'apprentissage.

5.2.5 Résultats expérimentaux

L'analyse précédente est maintenant illustrée à travers divers expériences au cours desquelles C varie ainsi que la taille de l'ensemble d'apprentissage. Pour toutes les expériences présentées, ε est fixé à 10^{-6}. Ainsi nous montrons quels sont les effets de comparaisons hâtives où ce paramètre n'est pas ajusté.

Comportement des méthodes pour une taille d'apprentissage croissante La figure 5.1 page 68 montre les résultats obtenus quand la taille de l'ensemble d'apprentissage varie et ce pour différents jeux de paramètres. Il est observé que :

- pour C donné, les résultats relatifs des différentes méthodes sont analogues en temps d'apprentissage indépendamment de la largeur de bande,
- la meilleure configuration pour CVM vis à vis du temps d'apprentissage est un C de grandeur intermédiaire, alors qu'en ce qui concerne la précision des résultats, une petite valeur de C est plus recommandée,
- pour un petit C, CVM est la méthode la plus performante en généralisation,
- pour un grand C, la performance en généralisation se détériore.

Ces résultats montrent clairement que les comportements caractéristiques des méthodes dépend des hyperparamètres et en particulier de C.

Comportement des méthodes en fonction de C La figure 5.2 page 69 montre comment l'apprentissage change avec l'ordre de grandeur de C pour chacune des implémentations étudiées. Les même expériences sont conduites sur deux tailles de bases d'apprentissage différentes afin d'illustrer l'effet amplificateur du nombre de points sur la sensibilité de CVM.

petit C SimpleSVM requiert trop de mémoire pour obtenir des résultats, libSVM est plus rapide grâce à l'effet de la condition d'arrêt plus faible et CVM est la plus efficace en généralisation (du fait de la régularisation supplémentaire introduite sur le noyau). Tous les points du *coreset* sont vecteurs supports, ce qui indique que le critère de sélection des points du *coreset* est bon.

grand C SimpleSVM est plus rapide que libSVM (qui est pénalisé par le critère d'arrêt strict) et tous deux sont aussi performants en test. CVM obtient la plus grande erreur en test et peu de points du *coreset* sont vecteurs supports. Par ailleurs on remarque une forte décorrélation entre le nombre de points du *coreset* et le nombre de vecteurs supports. L'explication est que les deux boucles imbriquées ont des objectifs contradictoires. La boucle interne de type ν-SVM produit une solution parcimonieuse et une marge très petite or dans le problème reformulé sous forme de MEB, plus la marge est petite et plus le rayon de la sphère est petit. Dans ce cas, la plupart des points tombent hors de la sphère et sont donc sélectionnés la boucle externe. Nous avons dont d'un côté une sélection intense de points et de l'autre une solution parcimonieuse.

Conclusion

Ce chapitre met en avant la prudence nécéssaire à la comparaison de méthodes de résolution. En effet, l'ordre de grandeur des hyper-paramètres est un facteur influent sur la rapidité de résolution. Cette influence est différente en fonction des algorithmes utilisés et il est intéressant de constater que plutôt que concurrentes, les méthodes de type SMO ou contraintes actives sont complémentaires. Il existe des « modes » de fonctionnement. Ainsi, pour des critères d'arrêt stricts (que ce soit dû directement à la valeur de ε ou bien à l'effet de la taille du problème et des hyper-paramètres) il vaut mieux utiliser une méthode de résolution de type contraintes actives. Cette observation de comportement des méthodes incite à réfléchir sur une approche hybride de résolution des SVM, apte à tirer parti des fonctionnements de chaque approche.

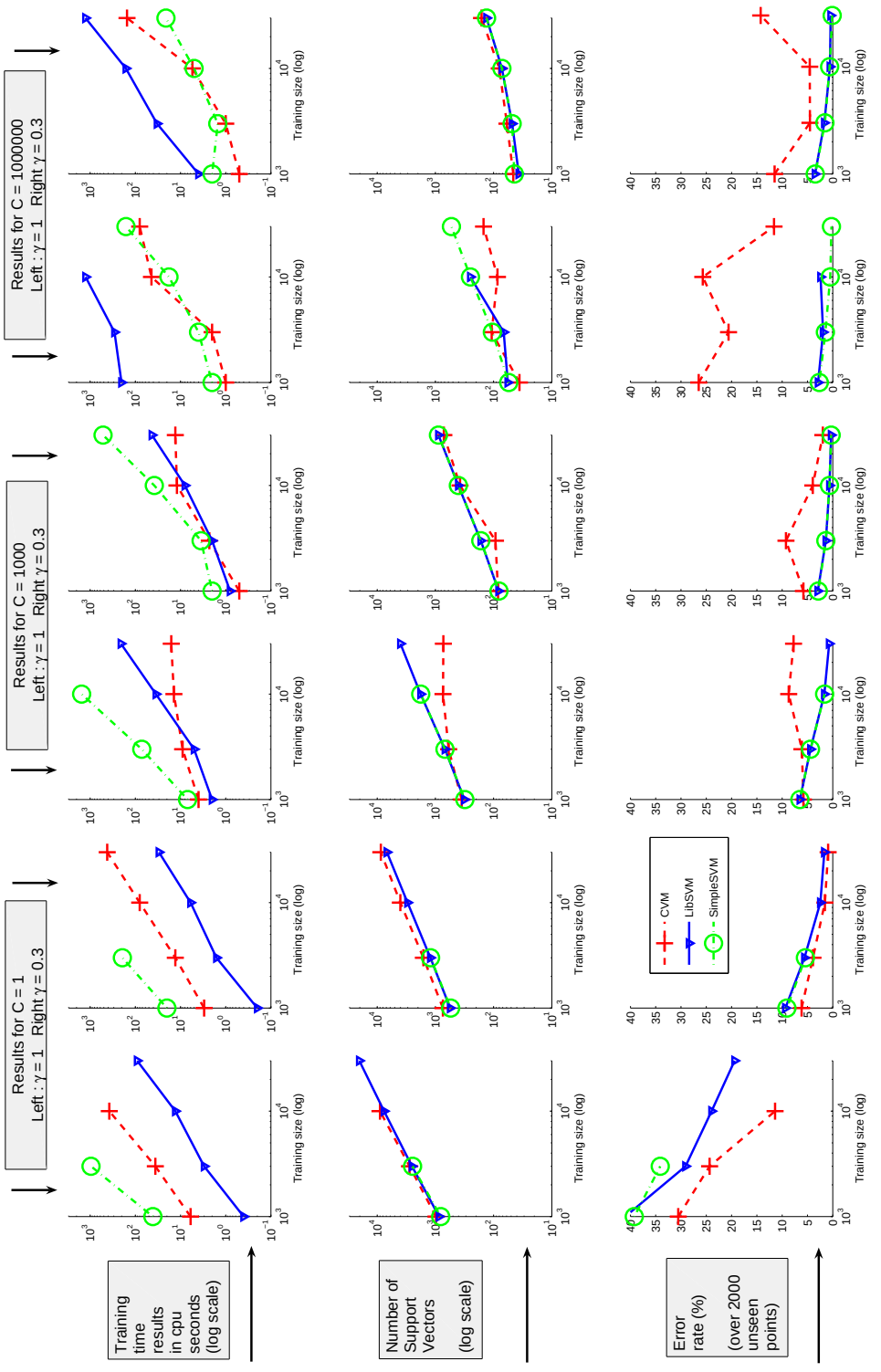


FIG. 5.1 : Comparatif pour différents jeux d'hyper-paramètres. Pour chaque colonne, la première ligne donne les temps d'apprentissage. La seconde ligne montre le nombre de vecteurs supports et la dernière ligne renseigne sur le taux d'erreur en test. Tous les résultats présentés sont obtenus avec $\varepsilon = 10^{-6}$.

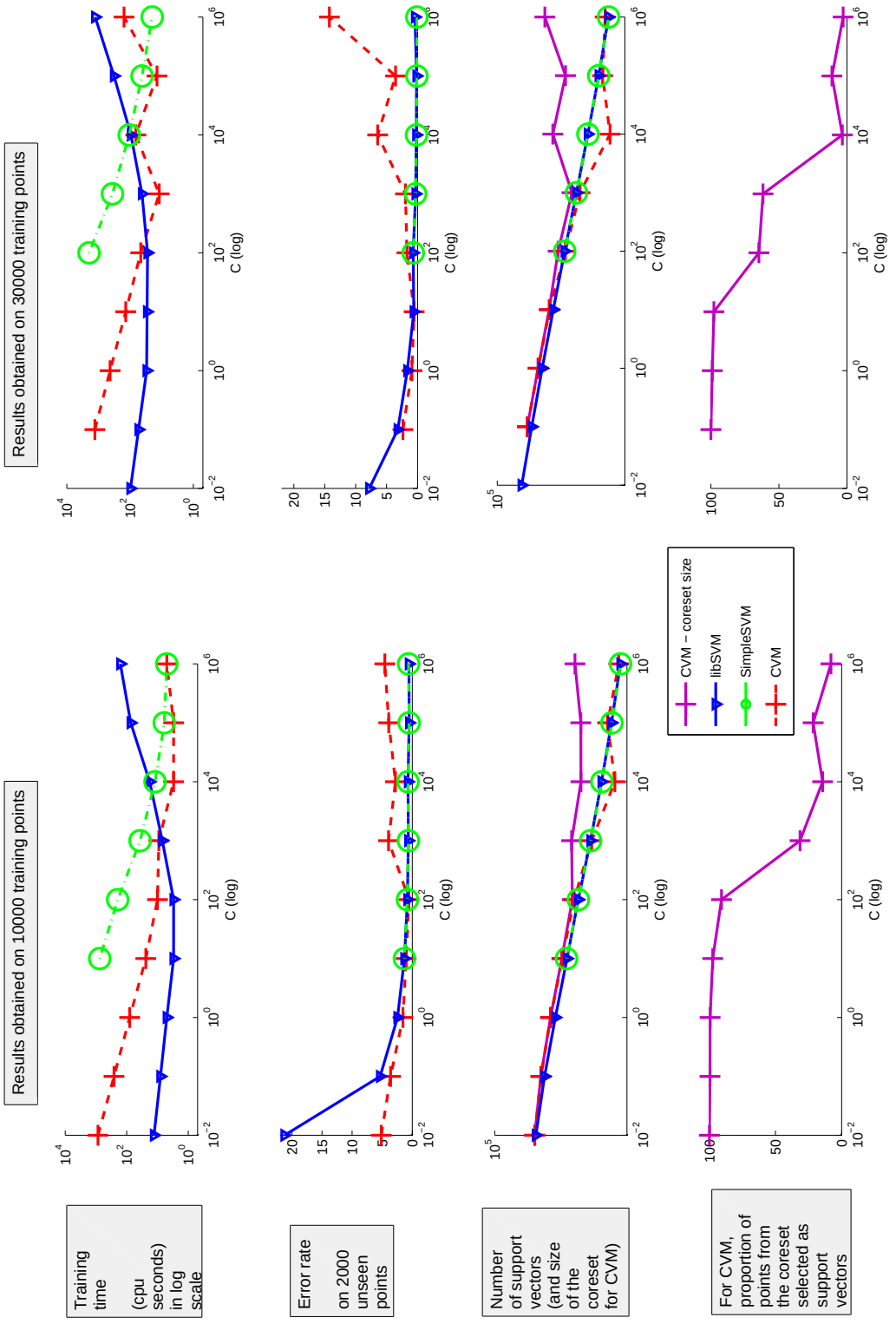


FIG. 5.2 : Expériences illustrant l'influence de l'hyper-paramètre C. La première colonne reporte les résultats obtenus sur une base de 10 000 points et la deuxième pour 30 000 points. Les figures du haut donnent le temps d'apprentissage (en log), la seconde ligne montre l'erreur en test sur 2000 points. La troisième ligne indique le nombre de vecteurs supports ainsi que la taille du *coreset*. La dernière ligne donne la proportion de vecteurs supports au sein du *coreset* pour CVM. Toutes les expériences sont faites avec $\gamma = 1$ et $\varepsilon = 10^{-6}$

6

Invariances et très grandes bases de données[1]

« Souvent, pour s'amuser, les hommes d'équipage
Prennent des albatros, vastes oiseaux des mers,
Qui suivent, indolents compagnons de voyage,
Le navire glissant sur les gouffres amers. »

Charles Baudelaire

Sommaire

6.1 De l'influence des invariances 72
6.2 De la formulation des invariances 73
6.3 Formulation unifiée 74
6.4 Des invariances dans la pratique pour la reconnaissance de caractères 75
6.5 Traitement par méthode hors ligne - SimpleSVM 79
6.6 Traitement par méthode en ligne - LASVM 80

SOUVENT, pour s'améliorer, les techniques d'apprentissage utilisent un *a priori*, les invariances, qui augmentent, par des variations des données, la capacité à reconnaître une forme. Les images sont un exemple illustratif par excellence, puisqu'il est en effet très naturel de reconnaître un objet déplacé ou tourné dans une image alors que c'est une tâche complexe pour une machine. Il existe beaucoup de méthodes pour essayer de résoudre ce problème [Grenander, 1993, Leen, 1995, Schölkopf et al., 1996, Simard et al., 1993, Wood, 1996]. Dans le cadre des méthodes à noyaux, il s'avère que traiter les invariances conduit en général à effectuer des pré-traitements sur les données et à générer des bases de données très grandes pour avoir des exemples de toutes les variations. Cette approche se heurte alors aux insuffisances de capacité des méthodes d'apprentissage à noyaux. Si la base de données n'est pas modifiée, alors c'est le noyau qui l'est et la conséquence directe est la limitation des types d'invariances que l'on peut gérer. Nous proposons dans ce chapitre deux approches efficaces qui relèvent de la catégorie « augmentation de la base de données » et montrons qu'avec un choix judicieux des variations à enregistrer et l'utilisation de l'apprentissage en ligne avec LASVM, nous traitons non seulement le problème des invariances mais nous aboutissons également à un SVM de très grande taille avec plusieurs millions de points.

[1] Les travaux contenus dans ce chapitre ont donné lieu aux publications [Loosli et al., 2005a, 2006a,b]

6.1 De l'influence des invariances

Décrivons tout d'abord pourquoi nous avons besoin de nous occuper des invariances sur un exemple jouet. Prenons des points en deux dimensions. Nous savons qu'il existe une incertitude sur les mesures de ces points, se traduisant par une rotation centrale du plan. Nous illustrons cela sur la figure 6.1 : l'image *(a)* montre les points mesurés et la classification correspondante. L'image *(b)* montre la trajectoire de chaque point en tenant compte de l'information sur les mesures. On voit alors que la classification effectuée précédemment n'est plus si bonne. Toutefois, l'image *(c)* montre qu'il existe une frontière qui classe bien à la fois les points mesurés et leurs variations. Cette nouvelle frontière montre que l'on a besoin soit de savoir classer les trajectoires des points, soit de savoir ajouter des points virtuels qui aideront à tenir compte de l'information *a priori* dont on dispose (image *(d)*).

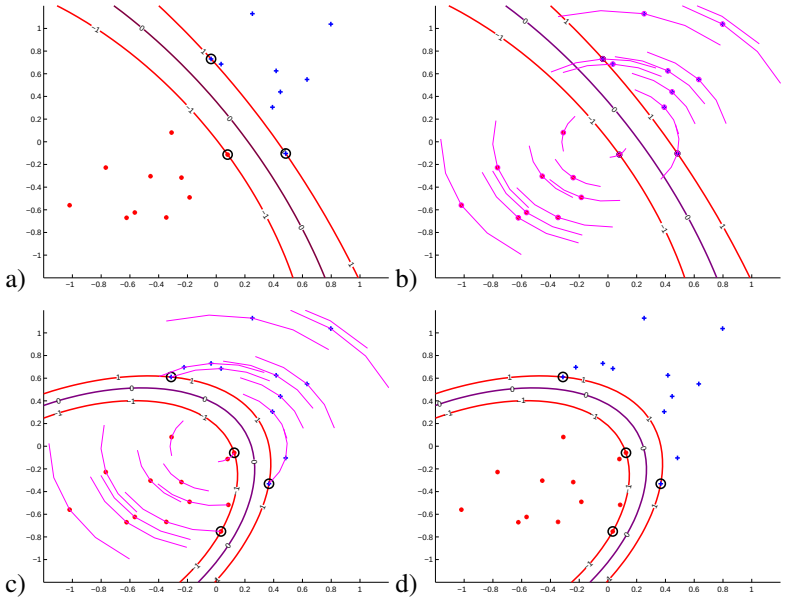


FIG. 6.1 : Ces figures illustrent l'influence des variations des points sur la frontière de décision pour un exemple jouet. L'image *(a)* montre les points mesurés et la classification correspondante. L'image *(b)* montre la trajectoire de chaque point en tenant compte de l'information sur les mesures. L'image *(c)* montre qu'il existe une frontière qui classe bien à la fois les points mesurés et leurs variations. L'image *(d)* tient compte de l'information *a priori* dont on dispose.

Ce simple exemple donne une bonne idée de la complexité du problème. Apprendre les trajectoires amène à des problèmes trop complexes [Graepel and Herbrich, 2004] et ajouter des points virtuels requière beaucoup de mémoire ($n \times t$ avec t le nombre de transformations par point). Toutefois, pour une application réaliste, nous avons besoin d'une méthode peu complexe et qui ne stocke pas trop d'information supplémentaire inutile. En effet seules quelques transformations sont réellement utiles à la classification, les autres ne font qu'alourdir le système sans rien apporter.

Après avoir donné une présentation plus formelle de l'intégration des invariances dans le problème des SVM, nous illustrons nos approches sur les bases d'OCR MNIST [LeCun et al., 1998] et USPS (voir annexe 7.3.5 page 112). Nous choisissons ces bases pour les possibilités de comparaison à d'autres méthodes et aussi pour les capacités d'illustration du fait de travailler avec des images. Nous expliquons dans un premier temps comment nous avons incorporé les transformations de manière pratique et puis discuterons des résultats obtenus.

6.2 De la formulation des invariances

Nous proposons de redonner le cadre de formalisation des invariances en reconnaissance de formes. Soit $\mathbf{x} \in E$ et Θ l'espace des paramètres de transformation applicables à $\mathbf{x}$. Une transformation de forme est une application T qui associe une forme transformée $T(\mathbf{x},\theta)$ à une forme $\mathbf{x}$ et des paramètres de transformation $\theta \in \Theta$:

$$\begin{array}{rccc} T: & E\times\Theta & \rightarrow & E \\ & \mathbf{x},\theta & \mapsto & T(\mathbf{x},\theta) \end{array}$$

De plus, on pose $T(\mathbf{x},0) = \mathbf{x}$.

Maintenant, si on considère un problème de classification, on défini la fonction de décision $\mathscr{D}(\mathbf{x})$ comme l'application de $\mathscr{X}^d$ dans $\{-1,1\}$ qui associe $\mathscr{D}(\mathbf{x})$ à $\mathbf{x}$. Si on veut que cette fonction de décision soit invariante, il faut qu'elle donne la même décision pour la forme et pour toutes ses transformations :

$$\mathscr{D}(T(x_i,0)) = \mathscr{D}(T(x_i,\theta))\ \forall\theta\in\Theta$$

Appliquées aux SVM dans le cas séparable, ces définitions donnent le problème primal suivant :

$$\left\{\begin{array}{l} \min\limits_{f,b}\ \dfrac{1}{2}\|f\|^2_{\mathscr{H}} \\ y_i\big(f(T(x_i,\theta))+b\big)\geq 1 \quad i\in[1,n],\theta\in\Theta \end{array}\right. \tag{6.1}$$

Nous pouvons en déduire la forme duale :

$$\left\{\begin{array}{ll} \max\limits_{\alpha} -\dfrac{1}{2}\displaystyle\sum_{i,j=1}^{n}\int\int_{\Theta}\alpha_i(\theta_1)\alpha_j(\theta_2)y_iy_jk(T(\mathbf{x}_i,\theta_1),T(\mathbf{x}_j,\theta_2))d\theta+\sum_{i=1}^{n}\int_{\Theta}\alpha_i(\theta)d\theta & \\ \text{avec} \quad \displaystyle\sum_{i=1}^{n}\int_{\Theta}\alpha_i(\theta)y_i d\theta = 0 & \\ \text{et} \quad \alpha_i(\theta)\geq 0 & i\in[1,...,n],\theta\in\Theta \end{array}\right. \tag{6.2}$$

Les multiplicateurs de Lagrange seront 0 pour tous les points à l'exception des vecteurs supports. De par la nature de l'espace de hypothèses, il est raisonnable d'estimer que $\alpha_i(\theta)$ aura des valeurs non nulles seulement pour un nombre fini de paramètres θ et nous pouvons récrire ce problème :

$$\left\{\begin{array}{ll} \max\limits_{\alpha} -\dfrac{1}{2}\displaystyle\sum_{i,j,\theta_1,\theta_2}\alpha_i(\theta_1)\alpha_j(\theta_2)y_iy_jk(T(\mathbf{x}_i,\theta_1),T(\mathbf{x}_j,\theta_2))+\sum_{i,\theta}\alpha_i(\theta) & \\ \text{avec} \quad \displaystyle\sum_{i,\theta}\alpha_i(\theta)y_i = 0 & \\ \text{et} \quad \alpha_i(\theta)\geq 0 & i\in[1,...,n],\theta\in\Theta \end{array}\right. \tag{6.3}$$

$$\left\{\begin{array}{lll} \max\limits_{\gamma\in\mathbb{R}^{n\times p}} & -\frac{1}{2}\gamma' G\gamma+\mathbf{e}'\gamma & \\ \text{avec} & \gamma'\mathbf{y}=0 & \\ \text{et} & \gamma_i\geq 0 & i\in[1,...,np] \end{array}\right. \tag{6.4}$$

avec $\gamma = [\alpha_1(\theta);\alpha_2(\theta);\ldots;\alpha_n(\theta)]$, G est un matrice structurée en blocs définie par $G_{IJ} = y_iy_jK^{ij}$ avec $K^{ij}_{kl} = k(T(x_i,\theta_k),T(x_j,\theta_l))$ et p est la taille de Θ.

CAS NON SÉPARABLE : Dans le cas non séparable, nous faisons face à un autre problème. Ajouter un terme autorisant les erreurs nous donne en dernière condition du système 6.2 $0\leq\int_{\Theta}\alpha_i(\theta)d\theta\leq C$.

Cela est assez proche du cas séparable. Toutefois nous ne pouvons plus faire l'hypothèse qu'un nombre fini de multiplicateurs de Lagrange non nuls sera suffisant. En effet, si la trajectoire d'une transformation traverse une marge, alors il y a une infinité de multiplicateurs de Lagrange bornés tels que $\int_\Theta \alpha_i(\theta)d\theta = C$.

6.3 Formulation unifiée

On identifie trois principales approches dans les algorithmes traitant des invariances avec des méthodes à noyaux. Certaines sont basées sur un agrandissement artificiel de la base de données, d'autres reposent sur la modification de la fonction coût pour incorporer les invariances (et utilisent une métrique différente) et d'autres encore utilisent des approximations polynomiales pour représenter les trajectoires.

6.3.1 Agrandissement artificiel de la base d'apprentissage

La plus intuitive des approches pour apprendre les invariances est d'introduire dans la base d'apprentissage les variations à connaître. Ceci est fait soit avant de commencer à apprendre en générant des exemples existants déformés. La base d'apprentissage ainsi obtenue contient par conséquent la connaissance *a priori*. Au delà de la simplicité de mise en œuvre, cette approche est rapidement limitée dans la pratique par la taille des problèmes engendrés. *Virtual-SV.* Cette idée est appliquée par Schölkopf et al. [1996] avec le V-SV (vecteurs supports virtuels). Les auteurs proposent un moyen de réduire la taille du problème en partant de l'idée que toute l'information utile à la résolution du problème est contenue dans les vecteurs supports. Sous cette hypothèse, seules les transformations des vecteurs supports sont calculées et ajoutées à la base d'apprentissage. Cela suppose de faire un premier apprentissage sur la base d'origine afin de détecter les vecteurs supports, puis de calculer les transformations pour agrandir la base et enfin d'apprendre sur la base obtenue. En pratique, cette méthode donne des résultats comparables à ceux obtenus en transformant directement tous les exemples.

Notre formulation unifiée permet de résoudre le même problème en approximant $T(\mathbf{x}, \theta)$ par un nombre fini de points correspondants à un nombre fini de valeurs de θ fixées.

6.3.2 Adaptation de la distance aux invariances

Introduites en 1993 par Simard et al. [1993] et appelées les distances tangentes, la motivation première était de trouver une mesure plus adaptée que la distance Euclidienne pour traiter les invariances. Dans le domaine des algorithmes à vecteurs supports, l'idée a été reprise par plusieurs approches.

SVM Invariant Des vecteurs tangents sont associés à chaque exemple d'apprentissage pour être incorporés dans la fonction coût Chapelle and Schölkopf [2002]. L'optimisation de cette fonction coût équivaut à utiliser un SVM classique avec une première phase de préparation des données correspondant à une application linéaire. Dans le cas non-linéaire, les résultats sont similaires à l'exception de l'application qui est alors non linéaire.

Distance tangente et vecteurs tangents à noyaux (Tangent Vector Kernels) En suivant l'idée d'utiliser la distance tangente, des noyaux intégrant les invariances ont été proposés. Pozdnoukhov and Bengio [2003] proposent un noyau entre des trajectoires plutôt qu'entre des points et définissent une fonction qui donne une mesure de proximité entre un point et la trajectoire formée par les transformations d'un autre point.

Notre formulation unifiée est similaire à ces approches si les transformations sont approchées par un polynôme

du premier ordre ($T(\mathbf{x},\theta) \simeq \mathbf{x} + \nabla_\theta T(\mathbf{x},0)\theta$).

6.3.3 Approximations polynomiales

Semi-Definite Programming Machines.

En présentant les SDPM, Graepel and Herbrich [2004] essaient d'apprendre des données qui sont directement des trajectoires au lieu des habituels points. Le but est de séparer des trajectoire qui représentent des transformations différentiables des points de la base d'origine. Ce problème est résolu pour des transformations qui peuvent être soit représentées, soit approchées par des polynômes. L'approche reformule le problème sous la forme d'un programme semi-défini sous contraintes polynomiales. En revanche, cette approche n'est pas applicable sur de grands problèmes.

Notre formulation unifiée peut également contenir cette approche en donnant comme transformation un polynôme de second ordre ($T(\mathbf{x},\theta) \simeq \mathbf{x} + \nabla_\theta T(\mathbf{x},\theta)\theta + \frac{1}{2}\theta' H_\theta \theta$).

Dans le cas séparable, nous pouvons résoudre directement le problème et être ainsi plus efficace que les SDPM. Néanmoins, dans le cas non séparable il faut discrétiser l'espace des paramètres et SimpleSVM Invariant est alors pénalisé par rapport aux SDPM.

6.4 Des invariances dans la pratique pour la reconnaissance de caractères

Les invariances constituent un sujet déjà beaucoup étudié et il existe un grand nombre d'approches pour calculer les transformations Schölkopf et al. [1996], Simard et al. [1993], Wood [1996]. Ici nous décrivons brièvement comment nous combinons ces approches de manière à les appliquer à un algorithme avec le moins de mémoire possible.

6.4.1 Vecteurs tangents

Simard et al. [2000] expliquent comment utiliser l'algèbre de Lie et les vecteurs tangents pour appliquer des transformations affines arbitraires à des images. Une transformation affine peut être décrite comme la composition des six éléments de transformation : translation horizontales et verticales, rotation, transformations d'échelle horizontales et verticales et transformation hyperboliques. Pour chaque image, la méthode consiste à calculer un vecteur tangent pour chaque transformation élémentaire, c'est-à-dire la différence normalisée pixel à pixel entre une transformation infinitésimale de l'image et elle-même.

Les petites déformations affines peuvent facilement être approchées en ajoutant une combinaison linéaire de ces six vecteurs tangents élémentaires :

$$x_{affine}(i,j) = x(i,j) + \sum_{T \in \mathscr{T}} \tau_T t_T(i,j)$$

avec $\mathscr{T}$ l'ensemble des transformations élémentaires, $t_T(i,j)$ le vecteur tangent pour la transformation T et τ_T le coefficient pour la transformation T.

6.4.2 Champs de déformation

Il s'avère que les vecteurs tangents des six transformations affines élémentaires peuvent être dérivées à partir des vecteurs tangents $t_x(i,j)$ et $t_y(i,j)$, correspondants aux translations horizontales et verticales. Chaque transformation affine élémentaire est décrite comme un vecteur de champs.

En plus des transformations affines, nous considérons les transformations élastiques. L'idée est d'introduire des déformations locales aléatoires. Les champs de déformation sont utilisés pour changer les coordonnées d'un pixel. Les coefficients des champs de déformation sont générés aléatoirement et lissés par une gaussienne, $[f_x^T(i,j), f_y^T(i,j)]$ représentant la direction de déplacement de chaque pixel (i,j) lorsqu'une transformation infinitésimale est appliquée à une image. Le vecteur tangent pour la transformation T est alors :

$$t_T(i,j) = f_x^T(i,j)\,t_x(i,j) + f_y^T(i,j)\,t_y(i,j)$$

Cette propriété permet d'étendre l'approche des vecteurs tangents à des transformations arbitraires *élastiques*, c'est-à-dire à des transformations représentées par un champs de déformation $[\,f_x(i,j),\, f_y(i,j)\,]$. De petites transformations d'une image $x(i,j)$ sont alors facilement approchées par l'opération linéaire :

$$x_{def}(i,j) = x(i,j) + \tau * (f_x(i,j) * t_x(i,j) + f_y(i,j) * t_y(i,j))$$

Les champs de déformation plausibles sont facilement générés en tirant aléatoirement et indépendamment pour chaque pixel un vecteur de mouvement et en appliquant sur le résultat un filtre lissant. Les figure 6.2 montre des exemples de telles déformations. Les composantes horizontales et verticales sont générées indépendamment et peuvent ainsi être interchangées.

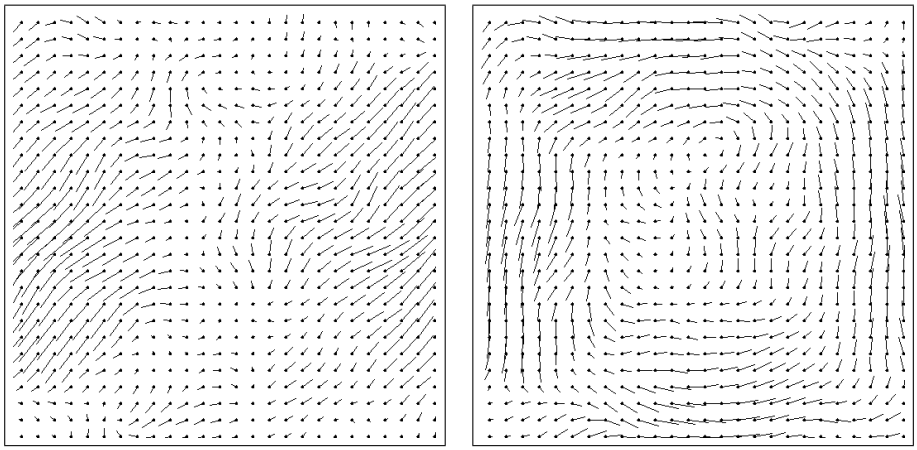


FIG. 6.2 : Cette figure montre deux exemples de champs de déformation. Sont représentées les combinaisons des champs horizontaux et verticaux. Le premier est un champs aléatoire lissé et le second une rotation modifiée par un bruit aléatoire. noise.

En plus des champs aléatoires, nous utilisons des champs réguliers et un champ nul. Ainsi nous pouvons obtenir des transformations affines et des transformations selon une seule direction. Enfin nous avons aussi généré des champs contrôlé, c'est-à-dire des champs réguliers (rotation par exemple) bruités.

Épaisseur des traits

Les vecteurs tangents horizontaux et verticaux peuvent aussi être utilisés pour obtenir la transformation d'épaississement des traits [Simard et al., 2000] qui permet aussi d'affiner les traits.

$$x_{thick}(i,j) = x(i,j) + \tau_b * \sqrt{t_x(i,j)^2 + t_y(i,j)^2},$$

avec τ_b le coefficient contrôlant la force de la transformation. Choisir $\tau_b < 0$ affine les traits et $\tau_b > 0$ les épaissit.

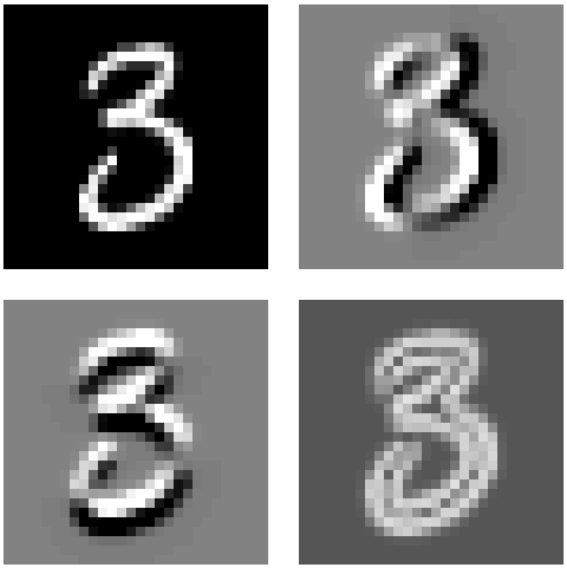


FIG. 6.3 : Cette figure montre le chiffre d'origine, les deux vecteurs tangents horizontaux et verticaux et le vecteur tangent d'épaississement

La base d'apprentissage virtuellement infinie

Il n'est pas possible de garder en mémoire l'ensemble des exemples transformés. Toutefois il est possible de générer efficacement des transformations aléatoires en combinant les méthodes décrites ci-dessus :

$$\begin{aligned} x_{trans}(i,j) &= x(i,j) + \tau_x * f_x(i,j) * t_x(i,j) \\ &+ \tau_y * f_y(i,j) * t_y(i,j) \\ &+ \tau_b * \sqrt{t_x(i,j)^2 + t_y(i,j)^2} \end{aligned}$$

Seules les images d'origine $x(i,j)$ et leurs vecteurs tangents de translation $t_x(i,j)$ et $t_y(i,j)$ sont mémorisés (voir figure 6.4).

De même, un ensemble de champs de déformation, pouvant être aussi bien utilisés en tant que $f_x(i,j)$ ou $f_y(i,j)$ sont pré-générés. Les coefficients τ_x, τ_y et τ_b permettent d'ajouter de la variabilité dans les transformations.

La figure 6.5 montre des exemples de toutes les transformations combinées. Il est possible de générer ainsi autant d'exemples que nécessaire en jouant sur les champs de déformation et les coefficients.

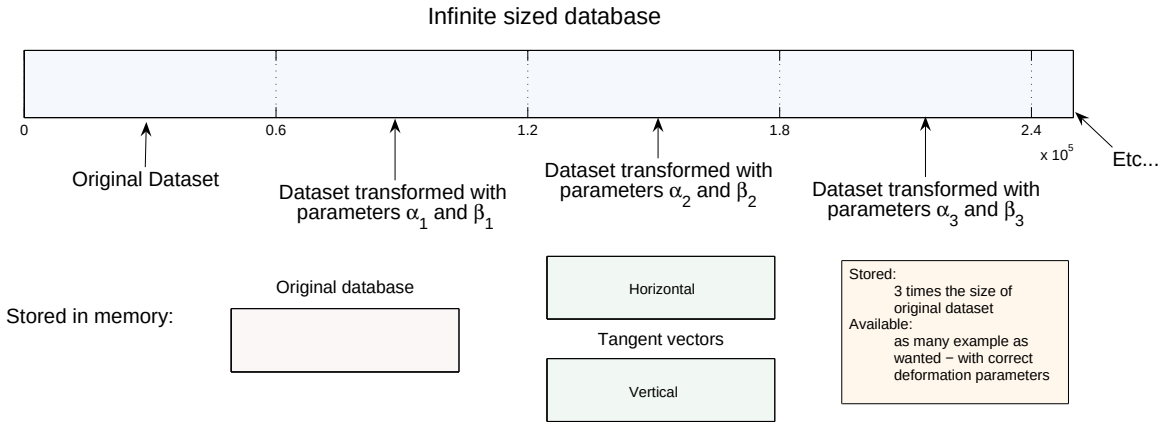


FIG. 6.4 : Illustration de la gestion de la base de données de taille infinie

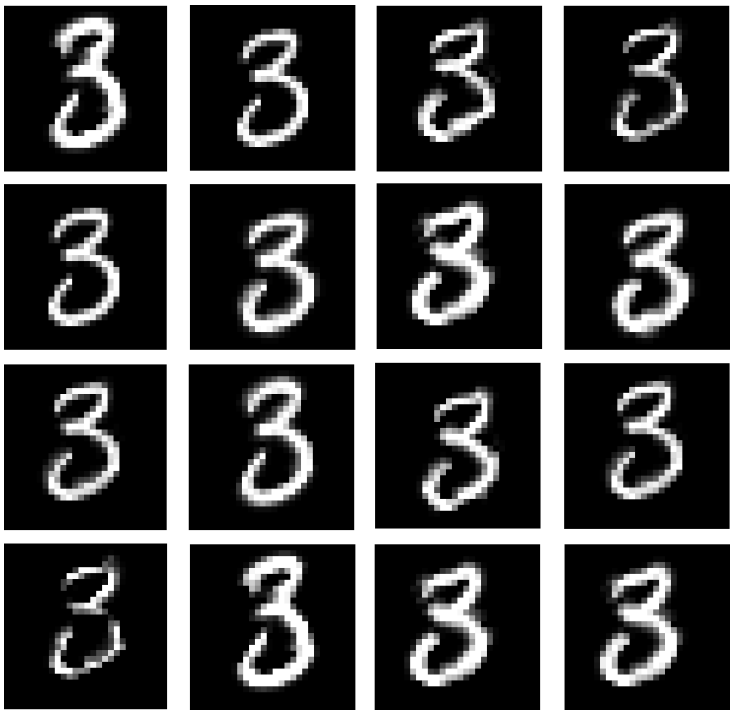


FIG. 6.5 : Cette figure donne 16 transformations d'un chiffre avec toutes les transformations précitées

Grandes translations

Toutes les transformations citées sont des transformations sous-pixels. Bien que la base MNIST contienne des chiffres centrés sur l'image, les expériences indiquent qu'il est toujours nécessaire de prendre en plus des grandes translations, de l'ordre de un ou deux pixels. Ainsi des déformations de un ou deux pixel dans une direction aléatoire sont également implémentées.

6.5 Traitement par méthode hors ligne - SimpleSVM

6.5.1 SimpleSVM Invariant

SimpleSVM Invariant peut intégrer des invariances telles que les vecteurs virtuels, les polynômes de premier ordre, etc. Dans l'implémentation présentée ici nous avons choisi d'utiliser l'approche de vecteurs virtuels. L'idée est de calculer au fur et à mesure de l'apprentissage les transformations des vecteurs candidats à être vecteurs supports et de n'ajouter à la base que les transformations qui deviennent vecteurs supports. De cette façon, il est possible de limiter l'agrandissement de la base d'apprentissage tout en n'effectuant qu'un seul apprentissage.

Algorithme 11 : *SimpleSVM Invariant*

1: $(I_w, I_0, I_C) \leftarrow$ initialiser
2: $(\alpha, \mu) \leftarrow$ résoudre le système sans contraintes(I_w)
3: **tant que** le solution n'est pas optimale **faire**
4: **si** $\min \alpha_w < 0$ ou $\max \alpha_w > C$ **alors**
5: projeter α_w à l'intérieur du domaine admissible
6: transférer le point associé de I_w vers I_0 ou I_C
7: **sinon**
8: calculer les multiplicateurs de Lagrange $\eta(I_C)$ et $\vartheta(I_0)$
9: **si** $\min \vartheta(I_0) < \varepsilon$ ou $\min \eta(I_C) < \varepsilon$ **alors**
10: calculer les transformations du points associé
11: transférer un point associé (ou la meilleure transformation) de I_0 ou I_C vers I_w
12: **fin si**
13: **fin si**
14: **fin tant que**

Nous proposons plusieurs heuristiques en fonction de la façon dont les transformations sont choisies.
- recherche complète : chaque point est candidat successivement et toutes les transformations sont examinées,
- recherche partielle : le point candidat est choisit dans un sous ensemble de point et seules les transformations de ce point sont considérées,
- recherche aléatoire : applicable aux deux précédantes variantes, cette approche consiste à tirer au hasard un sous ensemble de transformations plutôt que de toutes les vérifier.

6.5.2 Application à la reconnaissance de caractères

Il est connu qu'utiliser les invariances améliore les capacités de reconnaissance d'écriture manuscrite. Ici nous présentons les résultats obtenus sur la base USPS (annexe 7.3.5 page 112), pour laquelle il existe déjà de nombreux résultats afin de pouvoir comparer l'efficacité de notre méthode.

Plusieurs objectifs sont visés par ces expériences. Le premier concerne l'efficacité de la méthode proposée, le second est d'explorer les capacités des diverses heuristiques et de différentes combinaisons de transformations (on peut noter que la plupart des résultats publiés ne concernent que les translations de 1 pixel). Les résultats sont donnés dans le tableau 6.1 et les hyper-paramètres sont obtenus par validation croisée. Le tableau 6.2 rappelle les principaux résultats publiés pour USPS.

Noyau	Largeur de bande	C	Transformations	Erreur	Temps (app. et test)
Poly 5	0.1	10^-5	aucune	4.09	235 sec
Poly 5	0.1	10^-4	tr	3.44	1800 sec
Poly 5	0.1	10^-4	tr+rot	3.19	3200 sec
Poly 5	0.1	10^-4	tr+er	3.14	-
Poly 5	0.1	10^-4	tr+er+dil	3.24	2400 sec
Poly 5	*0.1*	10^-4	*tr+rot+er*	*2.99*	*2300 sec*
Poly 5	0.1	10^-4	tr+rot+er+dil	3.24	4800 sec

TAB. 6.1 : Résultats pour USPS : Les transformations appliquées sont des translations de 1 pixel (*tr*), des rotations (*rot*),l'affinage et la dilatation des traits (respectivement *er* et *dil*)

Méthode	Erreur
Vecteur Tangent et Rep. Local. Keysers et al. [2002]	2.0 %
Virtual SVM Schölkopf et al. [1996]	3.2 %
Hyperplan invariant + V-SV	3.0 %
SimpleSVM Invariant	3.0 %
Performance humaine	2.5 %

TAB. 6.2 : Quelques résultats publiés sur USPS

6.5.3 Discussion

Nous avons montré ici que le SimpleSVM Invariant est efficace. Il faut noter que nous avons implémenté les transformations en prenant le parti de discrétiser l'espace des paramètres, ce qui est équivalent au V-SV. Toutefois, nous obtenons de meilleurs résultats, ce qui peut être expliqué par une plus grande flexibilité de notre approche (les transformations ne sont pas limitées aux seuls vecteurs supports) et nous pouvons aussi prendre en compte plus de transformations.

6.6 Traitement par méthode en ligne - LASVM

Nous avons appliqué nos transformations à MNIST (annexe 7.3.5 page 112), en essayant de faire un compromis entre temps de calcul de usage mémoire. Nous avons donc stocké les vecteurs tangents et des champs de déformation, mais toutes les transformations sont calculées à la demande en tirant au hasard des champs de déformation stockés et des coefficients de force. Par ailleurs nous avons mis en place une stratégie de cache pour les exemples transformés ainsi que pour le noyau.

6.6.1 Du réglage des déformations

En théorie, plus on prend en compte des déformations et meilleur on est. Toutefois on risque d'introduire du bruit dans le cas de MNIST, compte tenu du fait que l'on ne cherchera pas à déformer les exemples de test et que la base est relativement régulière. Nous avons donc testé quelles déformations étaient les plus adaptées à notre problème en utilisant un jeu de validation. Nous avons par la même occasion fait une validation croisée sur le réglage du noyau. Les paramètres de la validation croisée sont les coefficients de force de déformation τ, la largeur

de bande γ du noyau que l'on choisit rbf (radial basis function). La figure 6.6.1 donne l'erreur en validation en fonction de τ pour le meilleur noyau, appris sur 5000 points et 10 transformations par point. On voit que changer l'épaisseur des traits n'est pas une bonne approche pour cette base. De la même manière, on peut se contenter des translations de 1 pixel au lieu de 2.

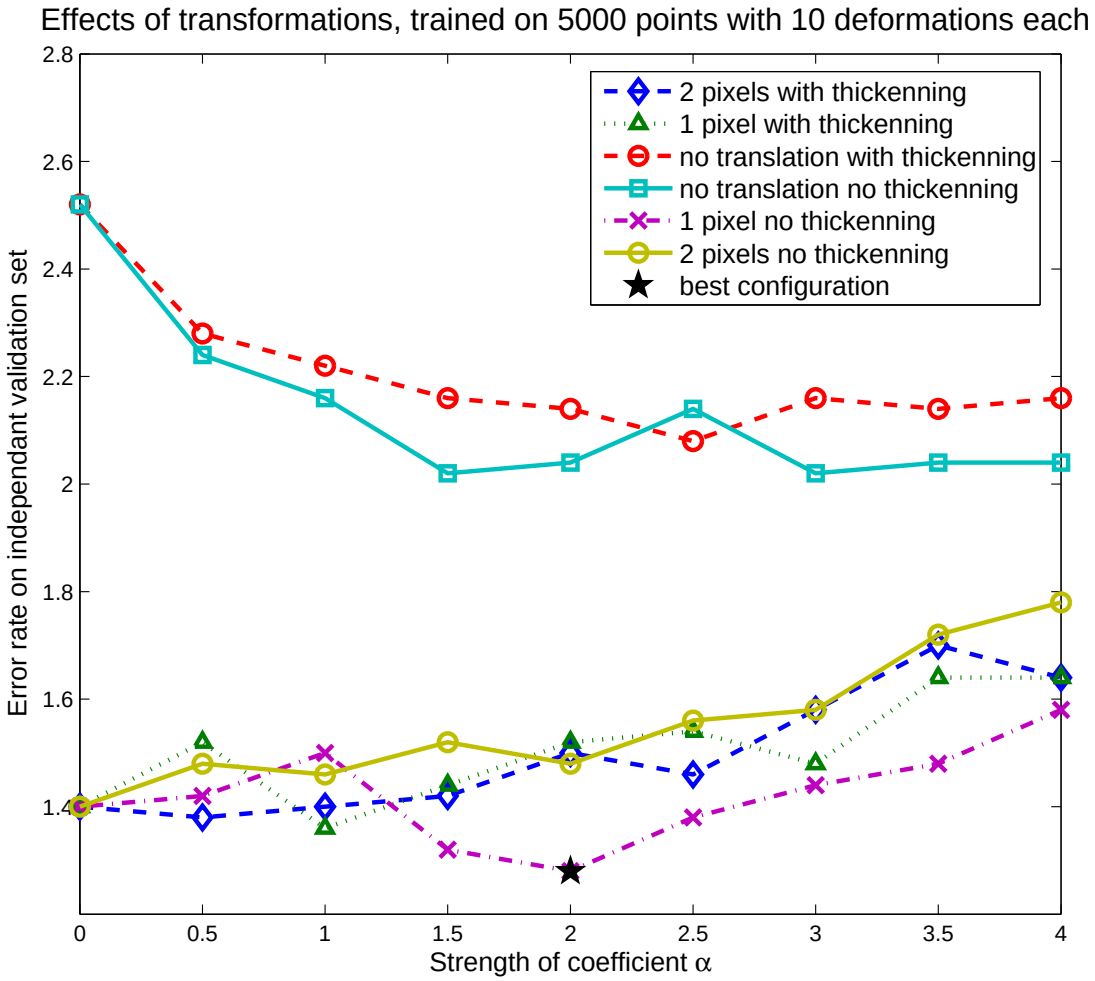


FIG. 6.6 : Effet des transformations sur la performance. Cette figure est obtenue sur un jeu de validation de 10000 points et appris avec 5000 points d'apprentissage, chacun déformé 10 fois (soit 55000 points en tout). Le noyau rbf a une largeur de bande $\gamma = 0.006$ obtenue par validation croisée. La meilleure configuration est obtenue pour les déformations élastiques de force $\tau = 2$ sans épaississement des traits et avec une translation possible de 1 pixel. La performance obtenue est de 1.28% et notons que le résultat sans transformation (obtenu pour $\tau = 0$, sans épaississement ni translation) est 2.52%

6.6.2 Du mode de sélection

La taille de la solution est un facteur de limitation. Nous cherchons donc le critère de sélection qui donnera les meilleurs résultats avec un bon compromis de parcimonie. La figure 6.7 page suivante reporte les résultats qui nous ont conduit à choisir le mode autoactif. En effet, les modes de sélection aléatoire (force brute) et autoactif donnent les meilleurs performances en test (premier graphique) tandis qu'entre ces deux modes, autoactif donne la solution la plus parcimonieuse (deuxième graphique). Nous garderons donc le mode autoactif.

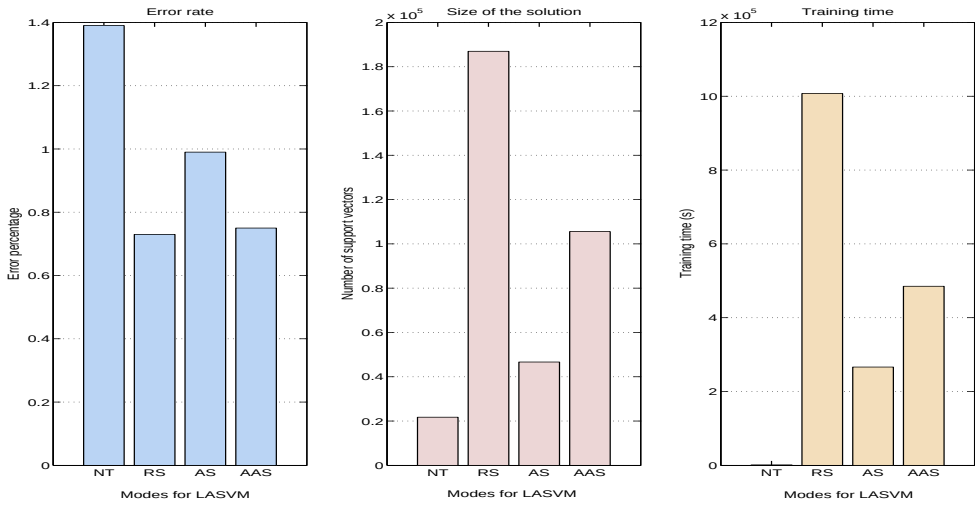


FIG. 6.7 : Ces graphiques comparent la performance (à gauche), la taille de la solution (centre) et le temps d'apprentissage (droite) pour différents modes de sélection de LASVM. La première colonne de chaque graphique reporte les résultats obtenus sans invariances, soit sur la base de donnée originale de 60 000 points. Les autres colonnes donnent respectivement les résultats pour les mode aléatoire (force brute) actif et autoactif. Ces expériences sont conduites sur 100 transformations par point, soit 600 000 points et selon les réglages obtenus précédemment. La sélection autoactive offre un bon compromis entre vitesse et performance

6.6.3 Des effets de l'optimisation complète

L'étape d'optimisation complète de LASVM est la dernière étape de l'algorithme. Dans notre cas, il n'y a pas de dernier point de la base puisque l'on peut toujours en générer. Dans un premier temps nous nous sommes contentés d'éliminer cette étape. Nous avons remarqué qu'après un grand nombre de points d'apprentissage, le nombre de vecteurs supports diminuait. Il s'avère qu'après un certain temps, les nouveaux exemples présentés n'apportent plus d'information au classifieur, il n'y a donc plus d'étape de *process* qui ajoute des points à la solution, mais il y a toujours l'étape de *reprocess* qui continue à optimiser et à enlever des points. Dans un deuxième temps nous avons introduit une étape d'optimisation complète régulièrement au cours de l'apprentissage (tous les 600 000 points). Nous avons observé le même phénomène mais plus tôt dans l'apprentissage. Se pose alors la question de la quantité d'optimisation nécessaire à chaque étape : est-il utile de moduler le nombre de *reprocess* par itération ?

6.6.4 Du nombre de reprocess

Le réglage de l'alternance entre *process* et *reprocess* est un point qui peut probablement améliorer les performance de LASVM. Le tableau 6.3 page suivante montre des résultats sur la qualité de la solution trouvée en fonction du nombre de *reprocess* effectué à chaque itération. Les résultats notés *nR/1P* désignent les expériences où *n reprocess* sont effectués à chaque fois qu'un *process* est fait. Les résultats notés *nR chaque* sont trouvés quand *n reprocess* sont fait à chaque fois qu'un nouveau point arrive, qu'il y ait ou non de phase de *process* pour lui. La principale conclusion de ces expériences est qu'il y a un compromis à faire, il ne faut ni trop optimiser à chaque étape, ni s'en tenir à un seul *reprocess*. Il y a donc matière à investigation pour ce point.

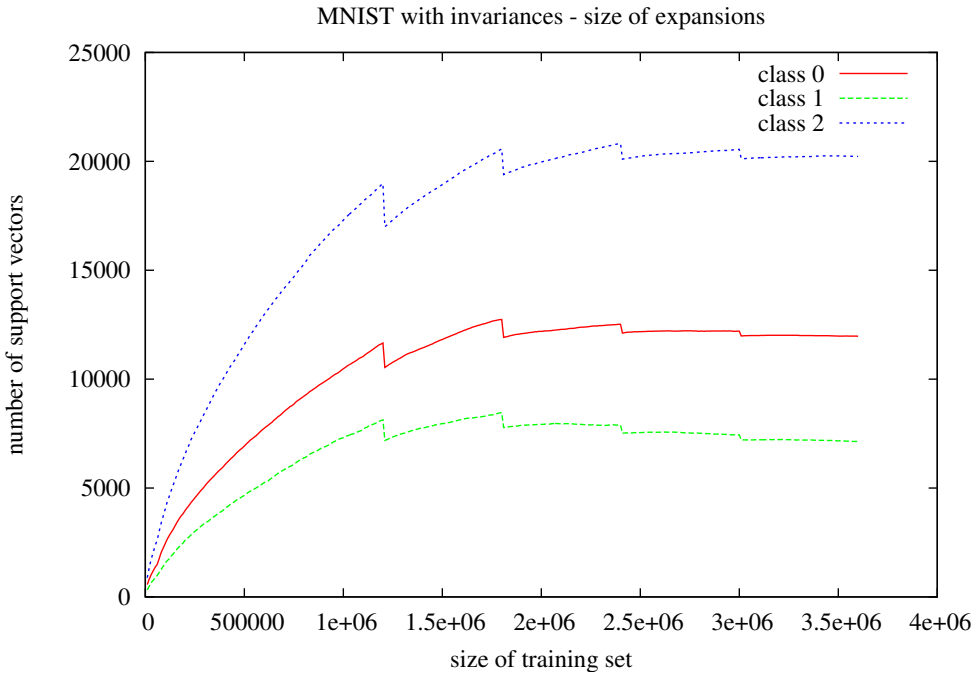


FIG. 6.8 : Cette figure montre l'évolution de la taille de la solution au cours de l'apprentissage. Ces résultats sont obtenus avec une sélection autoactive et en faisant régulièrement une phase de fin (optimisation complète). Chaque saut correspond à une phase d'optimisation complète. Nous remarquons ici que le nombre de vecteurs supports fini par décroître, ce qui peut être expliqué par la corrélation entre les exemples d'apprentissage

	1R / 1P	2R / 1P	3R / 1P	4R / 1P	5R / 1P	1R chaque	2R chaque	3R chaque
Taille max	24531	21903	21436	20588	20029	23891	21660	20596
Points enlevés	1861	1258	934	777	537	1487	548	221
Prop. d'enlevés	7.6%	5.7%	4.3%	3.7%	2.6%	6.2%	2.5%	1.0%
Temps d'app.(sec)	1621	1548	1511	1482	1441	1857	1753	1685
Taux d'erreur	2.13%	2.08%	2.19%	2.09%	2.07%	2.06%	2.17%	2.13%

TAB. 6.3 : Effets des transformations sur les performances. Le tableau montre la comparaison entre différents compromis *process/reprocess*. Le nombre de *reprocess* consécutifs après chaque *process* varie, et de la même façon après chaque point, qu'il y ait eu ou pas une phase de *process*

6.6.5 Des résultats globaux de LASVM pour les invariances sur MNIST

Cette expérience s'appuie sur l'hypothèse que la solution proposée s'améliore au fur et à mesure qu'elle prend en compte un nombre croissant de déformations. Le risque de cette approche est de perdre la capacité de reconnaitre les exemples non déformés. Pour vérifier le bon comportement de notre approche, nous avons vérifié au cours de l'apprentissage à la fois la performance en test et la performance en apprentissage sur la base d'origine. Nous avons conduite cette expérience pour la tâche « 2 contre tous » et le résultats est reporté figure 6.9. Nous vérifions donc expérimentallement que non seulement les performances s'accroissent dans le temps mais aussi que nous ne perdons pas la capacité de reconnaître les premiers exemples vus.

Le réglage pour les résultats finaux sont : noyau rbf avec une largeur de bande $\gamma = 0.006$ et $C = 1000$. Le coefficient de déformation est de $\tau = 2$. Nous ne touchons pas à l'épaississement des traits ($\tau_b = 0$). Le critère

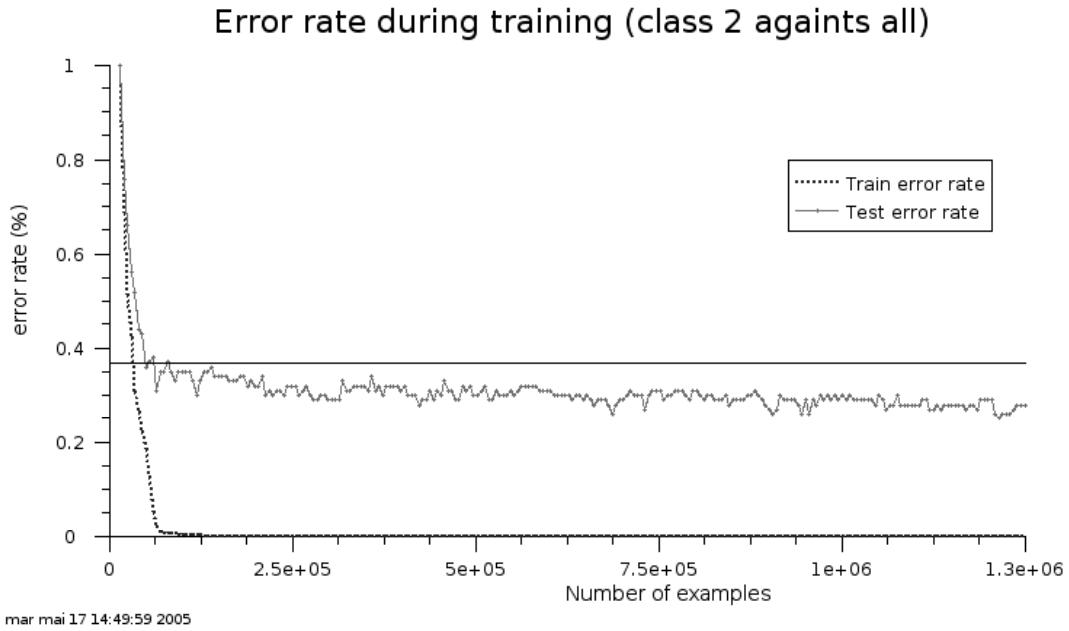


FIG. 6.9 : Cette figure montre l'évolution de la performance en apprentissage sur la base d'origine et de la performance en test au cours de l'apprentissage. Les résultats reportés ici correspondent au problème « 2 contre tous »

de sélection est le mode actif. Les meilleurs résultats obtenus sont 0.67%. De meilleures performances ont été publiées avec des réseaux à convolution ou en modifiant aussi le jeu de test Schölkopf and Smola [2002], Simard et al. [2003]. Cependant ces résultats sont aussi bons que les autres méthodes qui ne déforment en aucune façon les exemples de test. De plus, et c'est là notre principal propos, ils sont obtenus en entraînant 10 SVM binaires de 8 millions de points chacun en dimension 784. Le temps global d'apprentissage en test est de 8 jours sur une seule machine.

Conclusion

Au-delà du traitement des invariances, ce chapitre montre qu'il est possible de traiter avec des SVM des problèmes de très grande taille. L'approche privilégiée pour résoudre le passage à l'échelle est l'approche en ligne. Ces deux problèmes, l'apprentissage en ligne et l'apprentissage des grandes bases de données se confrontent à des problèmes similaires, à savoir le manque de mémoire et le temps de calcul. En y regardant de plus près, on s'aperçoit vite que résoudre le problème de la mémoire résoud une partie des problèmes de temps de calcul. Ainsi, utiliser un algorithme prévu pour l'utilisation en ligne à un problème de grande dimension est une approche non seulement raisonnable mais aussi efficace, comme le montrent les résultats obtenus sur les 8 millions d'exemples traités (en notant de plus que le problème traité contient dix classes et que la taille effective du problème résolu (avec l'approche du 1 contre tous) est de 80 millions de points).

7

Détection de changement [1]

« Celui qui copie la nature est impuissant,
celui qui l'interprète est ridicule,
celui qui l'ignore n'est rien du tout. »

René Barjavel

Sommaire

7.1 Familles exponentielles non paramétriques et méthodes à noyaux 86
7.2 Tests Statistiques et détection de rupture 86
7.3 Expérimentations 89

IL EXISTE beaucoup de travaux sur la segmentation de signaux du fait du grand nombre d'applications possibles, en reconnaissance vocale, indexation musicale ou détection de défauts par exemple. Ces travaux ont conduit à de nombreux algorithmes pour la détection de changement (voir Basseville and Nikiforov [1993] pour une revue complète). Jusqu'à maintenant les principales méthodes utilisent des modèles paramétriques de la distribution. Du côté des méthodes non paramétriques, plusieurs modèles ont été proposés (voir Markou and Singh [2003] pour une revue détaillée dans le contexte de la détection de nouveauté) tels que les réseaux de neurones [Fancourt and Principe, 2000], les modèles de Markov cachés, les modèles bayésiens [Roberts et al., 2004],les SVM [Davy and Godsill, 2002] (et autres méthodes à noyaux [Nguyen et al., 2005]) et les mesures de croyance [Lenser and Veloso, 2005].

Nous utlisons comme Desobry et al. [2005] le SVM à une classe pour effectuer un test statistique qui évalue la probabilité d'un changement abrupt à un instant donné dans les signaux. Toutefois notre approche n'utilise qu'un seul SVM au lieu de deux, diminuant ainsi le temps d'apprentissage.

[1] Ce chapitre est la synthèse des travaux [Loosli et al., 2005c,d,e]

7.1 Familles exponentielles non paramétriques et méthodes à noyaux

Nous introduisons ici brièvements les familles exponentielles non paramétriques qui nous permettent de faire le lien entre le SVM à une classe et les tests statistiques (voir Canu and Smola [2006]).

Notons $\mu(\mathscr{X})$ une mesure sur $\mathscr{X}$. Supposons par ailleurs qu'il existe un *RKHS* noté $\mathscr{H}$ avec le produit scalaire $\langle .,. \rangle_{\mathscr{H}}$ et le noyau reproduisant k tel $k(x,.)$ soit la statistique suffisante de x. Alors la densité de probabilité $\mathbb{P}(x;\theta)$ des familles exponentielles est donnée par :

$$\mathbb{P}(x;\theta) = \mu(x) exp\big(\langle \theta(.), k(x,.)\rangle_{\mathscr{H}} - g(\theta)\big) \tag{7.1}$$

avec $\theta \in \Theta \subseteq \mathbb{R}^p$ et

$$g(\theta) = log \int_{\mathscr{X}} exp\big(\langle \theta(.), k(x,.)\rangle_{\mathscr{H}}\big) d\mu(x). \tag{7.2}$$

si l'intégrale existe. Ainsi, le domaine Θ est défini comme l'ensemble pour lequel $g(\theta) < \infty$.

Le choix de la mesure μ est lié au noyau.

LEMME - NOYAU ADMISSIBLE : un noyau k est admissible pour la mesure μ si

$$\int_{\mathscr{X}} exp\big(k(x,x)^{1/2}\big) d\mu(x) < \infty$$

PREUVE : Selon l'équation 7.2, $g(\theta)$ est fini si

$$\int_{\mathscr{X}} exp\big(\langle \theta(.), k(x,.)\rangle_{\mathscr{H}}\big) d\mu(x) < \infty.$$

Or,

$$\begin{aligned} \int_{\mathscr{X}} exp\big(\langle \theta(.), k(x,.)\rangle\big) d\mu(x) &\leq \int_{\mathscr{X}} exp\big(\|\theta\|\|k\|\big)^{1/2} d\mu(x) \\ &\leq exp\big(\|\theta\|\big)^{1/2} \int_{\mathscr{X}} exp\big(\|k\|\big)^{1/2} d\mu(x) \\ &\leq exp\big(\|\theta\|\big)^{1/2} \int_{\mathscr{X}} exp\big(k(x,x)\big)^{1/2} d\mu(x) \end{aligned}$$

□

Ainsi le noyau gaussien n'est pas admissible avec la mesure de lebesgue mais on peut choisir une mesure pour lequel il l'est.

Ici θ est le paramètre naturel et $g(\theta)$ est la fonction de log-partition. Avec $\theta(x) = \langle \theta(.), k(x,.)\rangle_{\mathscr{H}}$, nous avons :

$$\mathbb{P}(x;\theta) = \mu(x) exp\big(\theta(x) - g(\theta)\big). \tag{7.3}$$

7.2 Tests Statistiques et détection de rupture

Soit $X_i, i = 1,2t$ une suite de variables aléatoires distribuées selon une distribution $\mathbb{P}_i$. Nous cherchons à savoir si un changement est arrivé au temps t. Pour commencer dans un cadre simple, nous supposons que la séquence est stationnaire de 1 à t et de $t+1$ à $2t$, c'est-à-dire qu'il existe des distributions $\mathbb{P}_0$ et $\mathbb{P}_1$ telles que $P_i = P_0, i \in [1,t]$ et $P_i = P_1, i \in [t+1,2t]$. Notre problème est de savoir si $\mathbb{P}_0 = \mathbb{P}_1$ (aucun changement) ou si au contraire $\mathbb{P}_0 \neq \mathbb{P}_1$ (un changement est arrivé). La reformulation statistique peut se faire comme suit :

$$\begin{cases} \mathbb{H}_0 & : \mathbb{P}_0 = \mathbb{P}_1 \\ \mathbb{H}_1 & : \mathbb{P}_0 \neq \mathbb{P}_1 \end{cases}$$

Notre objectif est d'avoir une méthode universelle qui fonctionne avec n'importe quelle densité. Nous faisons quelques approximations pour clarifier le cadre de travail. Tout d'abord nous considérons que les densités $\mathbb{P}_0$ et $\mathbb{P}_1$ appartiennent à la famille exponentielle généralisée et donc qu'il existe un espace de Hilbert à noyaux reproduisant $\mathbb{H}$ avec le produit scalaire $\langle .,. \rangle_{\mathbb{H}}$ et un noyau reproduisant k tel que [Smola, 2004] :

$$\mathbb{P}_0(x) = \mu(x)\exp\langle \theta_0(.), k(x,.)\rangle_{\mathscr{H}} - g(\theta_0),\ \mathbb{P}_1(x) = \mu(x)\exp\langle \theta_1(.), k(x,.)\rangle_{\mathscr{H}} - g(\theta_1)$$

avec μ une mesure de probabilité appelée le support et $g(\theta)$ la fonction de log-partition.

La seconde hypothèse est que les paramètres fonctionnels θ_0 et θ_1 de ces densités seront estimés sur les données, respectivement sur la première et la seconde moitié de l'échantillon considéré en utilisant un SVM à une classe. Ce faisant nous suivons notre hypothèse initiale, à savoir qu'avant le temps t nous savons que la distribution est constante et égale à $\mathbb{P}_0$. L'algorithme du SVM à une classe (OC-SVM - [Scholkopf et al., 2000]) nous donne une bonne estimation de cette densité. $\widehat{\mathbb{P}}_1(x)$ peut être vu come une approximation robuste de l'estimateur du maximum de vraisemblance. Utiliser le SVM à une classe et le modele des familles exponentielles s'écrit de la façon suivante :

$$\widehat{\mathbb{P}}_j(x) = \mu(x)\exp\left(\sum_{i=1}^{t} \alpha_i^{(j)} k(x,x_i) - g(\theta_0)\right) \quad j = 1,2$$

où $\alpha_i^{(0)}$ est obtenu en résolvant le SVM à une classe sur la première moitié des données (x_1 to x_t) tandis que $\alpha_i^{(1)}$ est obtenu de la même façon sur la seconde moitié des données x_{t+1} to x_{2t}). Forts de ces trois hypothèses, après simplification des calculs, la région d'acceptation d'un test de type « vraisemblance généralisé » (VG) est donnée par :

$$\sum_{j=t+1}^{2t}\left(\sum_{i=1}^{t} \alpha_i^{(0)} k(x_j,x_i)\right) < s''$$

Cela aboutit à l'algorithme de détection de nouveauté proposé dans Scholkopf et al. [2000]. La mise en œuvre de ce test peut se faire de plusieurs façons, selon l'interprétation que l'on en fait.

7.2.1 Tests sur la variance des sorties de l'OC-SVM

$$\mathbb{V}\mathrm{ar}_j\left(\sum_{i=t}^{t} \alpha_i^{(0)} k(x_j,x_i)\right) < s, \quad j \in [t+1, 2t]$$

Le seuil reste à déterminer. Nous utilisons une heuristique pour cela. Nous laissons à l'algorithme une période d'initialisation correspondant à quelques secondes de signaux de manière à observer l'ordre de grandeur moyenne et en déduire un seuil. Une autre heuristique utilisée consiste à travailler avec un seuil mobile (s) et un seuil minimal (s_m) en dessous duquel on considère que toute variation est du bruit). On utilise aussi un délai pendant lequel on stabilise s. Les règles utilisées sont les suivantes :

- s et s_m ont pour valeur initiale 0
- pendant la phase d'initialisation, s_m est égal à la moyenne des variances obtenues de puis le début,
- à la fin de la phase d'initialisation, s_m est fixé,
- si la valeur courante de s est inférieure à s_0 : alors $s = s_m$,
- si la variance à dépassé le seuil s : s prend la valeur maximale des dernières valeurs de la variance,
- si la valeur de s est stable depuis un temps supérieur au délai défini : alors s est égal à la moyenne entre les dernières valeurs des variances obtenues et lui-même, sans jamais être inférieur à s_m.

Cette heuristique fonctionne bien sous réserve de disposer d'une phase d'initialisation représentative des signaux à venir (par exemple un état stable de « non activité » est un bon critère pour déterminer ce qui peut être considéré comme du bruit).

7.2.2 Tests sur le taux de mauvaise classification

Nous venons de donner une heuristique pour utiliser une méthode dont le seuil de détection dépend des données. Afin d'éviter ces étapes nous proposons d'approximer le test par un critère qui permet d'utiliser un seuil facile à régler. Nous remplaçons la variance par la proportion de points non attribués à la classe courante. Ainsi nous pouvons fixer le seuil comme étant un pourcentage fixe.

7.2.3 Tests sur la somme cumulée des sorties de l'OC-SVM

Inspirée de la méthode CUSUM, nous avons aussi utilisé la sortie du SVM comme un score qui se cumule négativement. Quand l'accumulation atteint un seuil on prend une décision. On pose donc deux seuils, un négatif et un positif. Si le seuil négatif est atteint on décide qu'il n'y a pas eu de rupture, si le seuil positif est atteint on décide qu'il y a eu rupture. Si aucun des deux seuils n'est atteint on regarde le point suivant pour essayer de décider (voir figure 7.1 pour plus de détails).

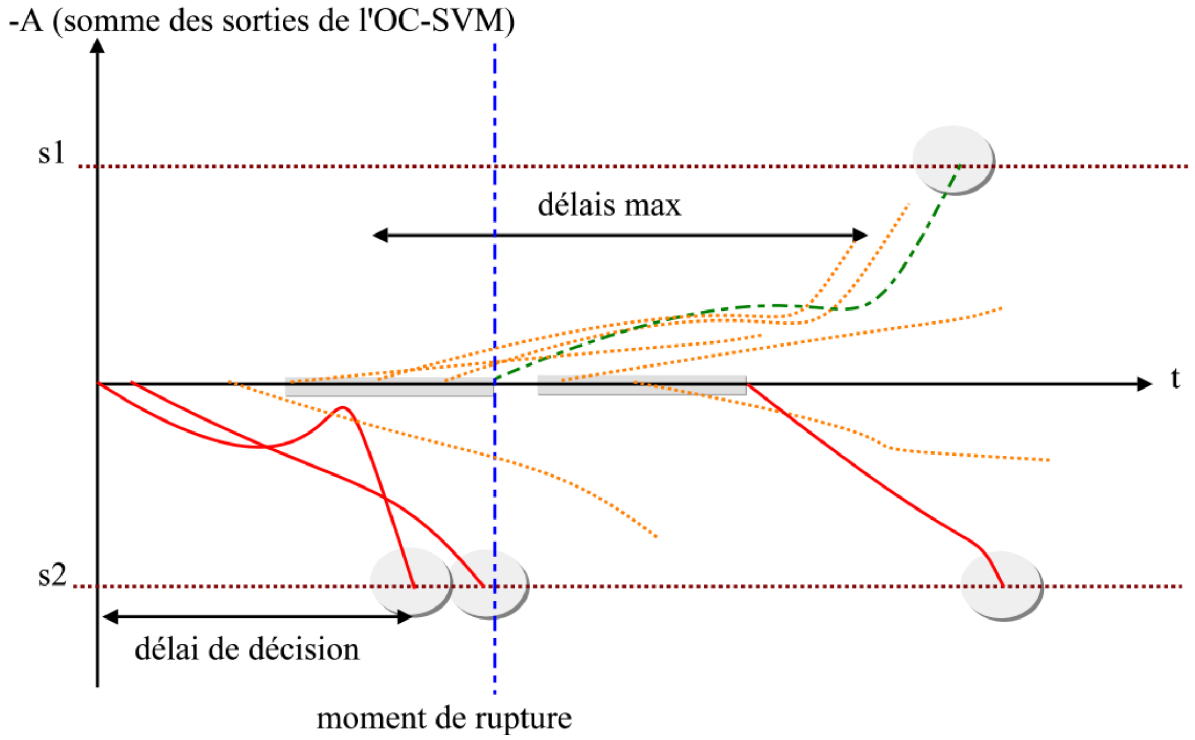


FIG. 7.1 : Illustration du fonctionnement de CUSUM. A chaque point, on regarde l'évolution de la somme de la sortie du SVM (concrètement, on classe le prochain point non vu et on ajoute sa sortie à la somme des points précédents). Trois événements arrêtent ce processus. 1/ on atteint le seuil de décision de stabilité de la classe (courbes pleines rouges). 2/ on atteint le seuil de décision de rupture (courbe verte en pointillés irréguliers). 3/ on atteint le délai maximal sans pouvoir prendre de décision (courbes oranges en pointillés).

7.3 Expérimentations

Dans cette partie nous allons d'abord donner quelques exemples sur des données synthétiques et sur des données utilisées dans notre pré-étude. Ensuite nous expliquerons notre démarche expérimentale et enfin nous donnerons les résultats obtenus pour deux jeux de données réelles [2].

7.3.1 Illustration de la méthode sur des données synthétiques

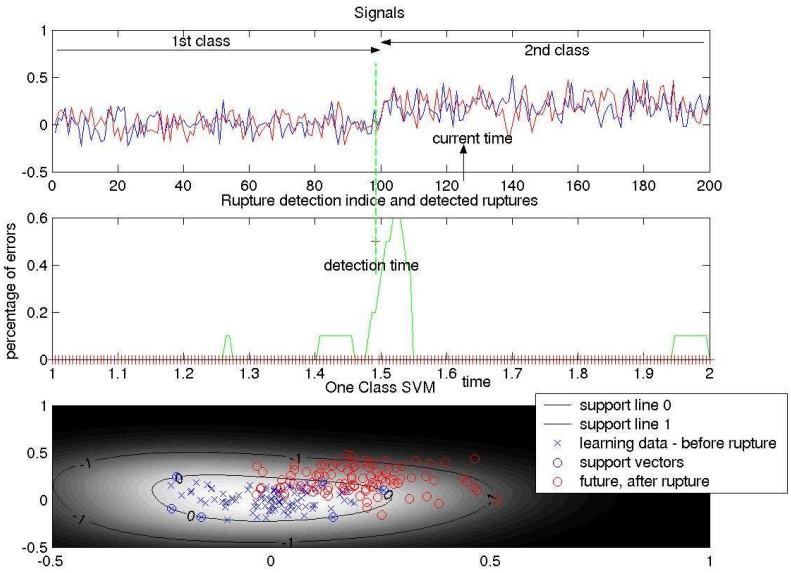


FIG. 7.2 : Cette figure montre le lien entre la détection de rupture et le SVM à une classe. La première partie montre les signaux issus de deux distributions gaussiennes. Le second graphe présente les ruptures trouvées par l'algorithme et la dernière partie est la sortie du SVM appris sur les données *passées*

Nous générons des données aléatoires distribuées selon des lois gaussiennes dont on fait varier les paramètres. Les données sont représentées par la moyenne et la variance sur une fenêtre mobile (figure 7.3.1 page suivante).

7.3.2 Illustration sur signaux réels

Nous avons testé notre méthode sur les signaux utilisés pour la première étude, à savoir les données issues d'accéléromètres pour la classification de mouvements.

Cette expérience (figure 7.4 page 91) montre qu'il est possible de segmenter utilement les données. Il n'y a pas de non détection sur des changements « à long terme » et les fausses alarmes seraient facilement décelées lors de la classification des tronçons de signaux. Pour deux des fausses alarmes (les deux dernières) on voit aussi qu'elles ont une caractéristique qui pourrait permettre de les éliminer d'office : elles apparaissent entre deux moments où la sommes cumulées est négative, ce qui n'est jamais le cas pour les bonne détections. Toutefois cette heuristique n'a pas été validée sur d'autres données jusqu'à présent.

[2] disponibles sur `http://asi.insa-rouen.fr/~gloosli/contextaware.html`

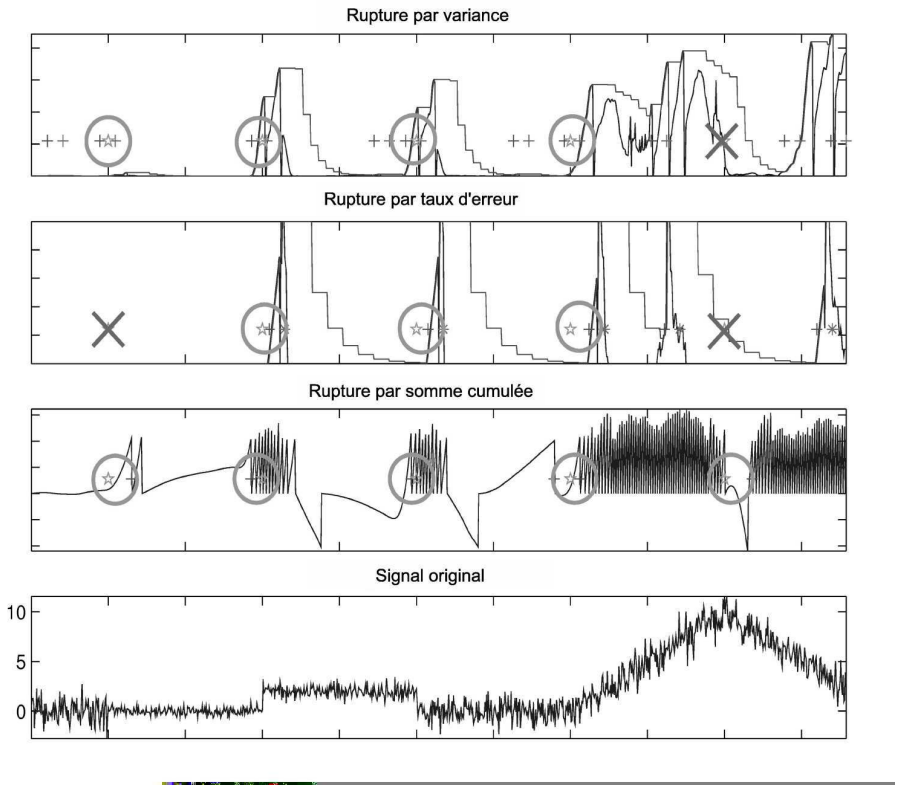


FIG. 7.3 : Exemple d'utilisation de l'algorithme sur des données issues de lois gaussiennes dont on fait varier la moyenne et la variance. La courbe du bas montre le signal de départ. Les trois premières montrent les résultats obtenus par les trois méthodes de détection de rupture.

7.3.3 Acquisition de signaux

Le ProComp est un système d'acquisition sur lequel nous disposons de cinq capteurs biologiques, à savoir un capteur de conductance de la peau, un capteur de respiration, de pression sanguine, un électromyogramme et un capteur de température périphérique (voir annexe 7.3.5 page 107).

Dans un cadre médical ou pour rechercher une information spécifique, les capteurs doivent être placés à des endroits soigneusement choisis et la personne équipée doit veiller à ne pas faire de mouvements qui perturbent les signaux. Dans notre approche nous cherchons à reproduire des données réalistes pour une utilisation dans la vie quotidienne. Par conséquent l'utilisateur ne doit pas « prendre garde » aux capteurs lors des prises de données. Dans le même ordre d'idée, les caractéristiques issues de signaux bruts sont généralistes et identiques pour tous les capteurs.

7.3.4 Utilisateur monitoré dans un contexte anodin

L'utilisateur est équipé des cinq capteurs biologiques ainsi que de trois accéléromètres tri-axiaux. Il peut ensuite aller et venir à sa convenance. Les données sont recueillies sur quelques minutes pendant lesquelles l'utilisateur est filmé. A partir de la vidéo, nous définissons à priori les moments où nous nous attendons à détecter un changement dans les signaux. Notre objectif est de déterminer à la fois si la détection automatique donne des

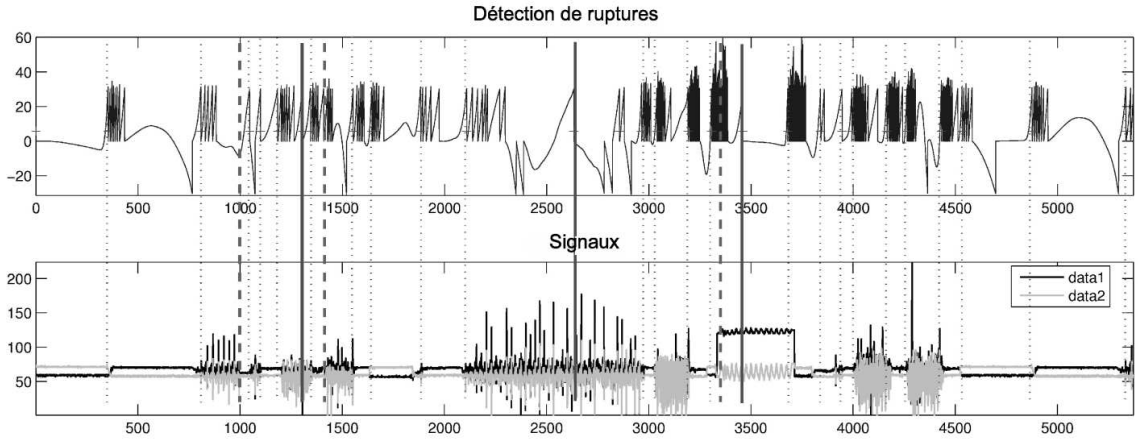


FIG. 7.4 : Résultats sur les données de mouvements avec la méthodes de somme cumulée. On remarque qu'il y a peu d'erreurs (bonnes détections en pointillés fin roses). Les non détections (traits épais pointillés verts) apparaissent à des moments ou les changements s'enchaînent rapidement. Les fausses alarmes sont notées en traits épais pleins roses.

résultats sensés mais aussi de voir jusqu'à quel point une observation extérieure est fiable.

Obtention de résultats

Les signaux obtenus sont traités « en ligne ». Il n'y a donc pas de normalisation globale possible. Par ailleurs les caractéristiques extraites sont la variance, la moyenne, le minimum, le maximum, et la transformée de Fourier sur une fenêtre glissante. La taille de la fenêtre est fixée arbitrairement de manière à la faire correspondre à une demi-seconde. La largeur de bande du noyau (gaussien) est obtenue par validation croisée.

Résultats et discussion

A posteriori	Manuel (à priori)	Auto (Variance)	Auto (Taux d'erreur)	Auto (Somme cumulée)
Nb de ruptures	16	30	23	17
Détection	16	15 (94%)	15(94%)	15(94%)
Fausse alarme	0	15 (50%)	8 (35%)	2 (12%)
Non Détection	1	2 (13%)	2 (13%)	2 (13%)

TAB. 7.1 : Résultats obtenus pour les différentes méthodes. Les ruptures détectées sont les mêmes mais les taux de fausse alarme diffèrent.

Les résultats présentés dans le tableau 7.1 valident notre approche. Entre les trois variantes la somme cumulée donne les résultats les moins bruités. Nous utiliserons donc préférablement cette méthode sur les données plus complexes (à savoir obtenues uniquement à partir de capteurs physiologiques).

7.3.5 Utilisateur monitoré dans le cadre d'un jeu vidéo

Pour cette expérience nous n'utilisons que les capteurs biologiques, car l'utilisateur reste assis (voir annexe 7.3.5 page 107).

Obtention de résultats

Les signaux, moins nombreux que précédemment, sont traités de façon similaire. La vidéo et les enregistrements des parties permettent au joueur de donner une liste d'événements significatifs pour lui, tandis qu'un observateur dresse lui aussi une liste d'événements. Ce double étiquetage a pour objectif de vérifier une de nos hypothèses de travail, à savoir qu'il est impossible, dans une situation réelle, de connaître avec certitude l'état que le système doit découvrir. On constate sur ce double étiquetage que le joueur a noté environ 20% d'événements significatifs en plus par rapport à l'observateur extérieur.

Résultats et discussion

A posteriori	Manuel Joueur (à priori)	Manuel Observateur (à priori)	Auto (Somme cumulée)
Nb de ruptures	50	39	55
Détection	50	39	36 - 32 (72% - 82%)
Fausse alarme	0	0	19 - 23 (35% - 42%)
Non Détection	0	11	14 - 7 (22% - 18%)

TAB. 7.2 : Résultats obtenus pour le joueur de jeux vidéos.

Les résultats (tableau 7.2) beaucoup plus mitigés sur cette expérience confirment le fait qu'avoir uniquement des données physiologiques est un problème plus complexe. Les accéléromètres fournissent en effet des informations « faciles » à segmenter, ce qui n'est pas le cas pour la conductivité de la peau (qui est lente) par exemple. Par ailleurs nous constatons que ce qui apparaît notable par l'utilisateur ou par l'observateur ne correspond pas nécessairement à une variation physiologique. En revanche nous détectons tout de même des changements qui correspondent au débuts de fins de parties ainsi qu'aux victoires ou défaites. Les événements plus intermédiaires comme blesser un adversaire ne semblent pas être détectables (et le plus souvent n'ont pas été notés par l'observateur).

Conclusion

L'utilisation de la formulation des méthodes à noyaux avec les familles exponentielles alliée au tests statistiques sur ces même familles exponentielles nous permet d'aboutir à un test non paramètriques que l'on met en pratique à l'aide d'un SVM à une classe. La mise à œuvre de ce test sur des données synthétiques nous permet d'évaluer différentes heuristiques de gestion du seuil de test. Enfin nous appliquons la méthode obtenue (proche du CUSUM) à des données réelles pour y retrouver les ruptures. Les données sont issues d'expériences filmées et sont annotées à l'aide de la vidéo. Nous sommes capables de retrouver la plus grande partie des ruptures observées par l'utilisateur ou par un observateur extérieur. Côté mise en œuvre, cette méthode bénéficie de la capacité de reprise à chaud de SimpleSVM car à chaque apprentissage, les données sont presque les mêmes.

Une fois les segments isolés dans les données, il reste à les identifier. Cette étape n'est pas effectuée dans le cadre des travaux présentés ici.

Conclusions et perspectives

« It was new. It was singular.
It was simple. It must succeed ! »

H. Nelson

Les travaux présentés au cours des chapitres de cette thèse rassemblent différentes approches au problème de l'apprentissage autonome et endurant. En partant du domaine applicatif de la détection de contexte, nous nous sommes confrontés aux limites des approches classiques de l'apprentissage statistique. Si un appareil tel qu'un téléphone portable était sensible à son contexte, cela sous-entendrait qu'il prend en compte sans cesse des informations sur le monde extérieur et sur les habitudes et préférences de l'utilisateur. Pour cela, il devrait donc apprendre en continu sinon en temps réel et travailler avec un nombre de données très important et toujours croissant. Les limites de l'apprentissage se trouvent précisément là. Comment apprendre en continu, de manière robuste, fiable et le tout sans disposer de grandes capacités de calcul et de mémoire ? Les travaux présentés ici s'attaquent à ces problématiques et nous avons choisi de travailler sur les SVM car cet algorithme a fait ses preuves en termes de performances en discrimination.

Permettre à des applications d'accéder à leur contexte ouvre un nombre important de perspectives dans l'interaction homme-machine. Les nombreux travaux visant à déterminer comment utiliser le contexte montrent le besoin de savoir récupérer ce contexte. Le fait que ces travaux se servent, la plupart du temps, d'une liste de contextes prédéfinis indique qu'il n'est pas nécessaire de connaître le contexte au sens large mais de savoir identifier ce qui est utile à l'application. C'est dans cette optique que nous avons défini la notion de contexte utile. Pour notre application cible, centrée sur la personne et son état et utilisant des capteurs physiologiques, nous définissons la notion d'état affectif. L'état affectif est le contexte utile lié à la personne qui utilise l'application, à ce qui l'affecte et donc affecte l'application. Nous faisons abstraction de l'application pour nous concentrer sur les moyens d'extraire, à partir des capteurs biologiques et physiques, des états affectifs. A cette fin nous avons défini une architecture en couche qui vise à organiser et simplifier les signaux bruts. Un des principaux avantages de cette approche est d'isoler des tâches relativement simples. De cet ensemble de tâches, nous n'avons mis en œuvre qu'une seule partie, qui est la détection de changements dans les signaux. En effet, en analysant les besoins en apprentissage automatique pour l'ensemble des tâches, nous nous sommes heurtés à un certain nombre de verrous scientifiques. Par conséquent, la plus grande partie des travaux présentés ici et effectués au cours de cette thèse concernent l'apprentissage d'un point de vue plus général avec comme objectif d'aboutir à une méthode d'apprentissage autonome et endurant.

Apprentissage autonome

Par autonome, nous entendons un apprentissage qui ne nécessite pas l'intervention d'un spécialiste voire même de l'utilisateur pour pouvoir fonctionner. Cela implique de faire appel à des méthodes aptes à se régler seules et en ligne. Par endurant, nous faisons référence à un usage réaliste des applications, c'est-à-dire un fonctionnement en temps réel, donc rapide et en ligne, pour un nombre très important de données et stable. Parce que les SVM donnent des résultats précis, nous avons focalisés l'ensemble de nos travaux sur cette méthode. Or, *a priori*, les SVM sont loin de répondre aux exigences d'un apprentissage autonome et endurant.

Les méthodes classiques d'optimisation, même les plus performantes, ne sont pas adaptées à la résolution des SVM si elles ne tiennent pas compte d'un des principaux atouts de cette méthode, à savoir la parcimonie de la solution. Ainsi les techniques de résolution les plus efficaces ont été conçues pour utiliser cette parcimonie. La méthode SMO est la plus utilisée. La méthode SimpleSVM que nous avons développée se montre compétitive avec SMO. Plus précisément, il s'avère que ces deux approches sont complémentaires.

Nous montrons en particulier que les différentes techniques de résolution des SVM ont des « modes » de fonctionnement, principalement dépendants des hyper-paramètres. SMO est pénalisé dans les cas très peu régularisés ou encore pour un critère d'arrêt très stricte alors que SimpleSVM y est presque insensible. En revanche, SimpleSVM est pénalisé dans les cas avec beaucoup de mélange ou très régularisés à cause du nombre de vecteurs supports tandis que SMO se comporte beaucoup mieux. Concernant SimpleSVM, nous avons développé une boîte à outils disponible sur Internet. Elle est codée en Matlab et est à notre connaissance la seule dans ce langage mettant en œuvre une technique de cache des valeurs du noyaux. Par ailleurs elle est conçue pour que l'intégration de nouvelles dérivations soit simplifiée.

L'observation des modes de fonctionnement, favorables ou non mais surtout complémentaires font envisager une perspective de travail : une méthode hybride capable de s'adapter en fonction des hyper-paramètres, qui puisse en quelque sorte piocher dans les atouts de SMO et de SimpleSVM. Dans une perspective à plus court terme, nous prévoyons d'intégrer à SimpleSVM les heuristiques courantes telles que le shrinking ou la sélection aléatoire des exemples à insérer dans la solution. Dans le même ordre d'idée, nous voulons évaluer les capacités de SimpleSVM si nous choisissons a priori *un nombre maximum de vecteurs supports.*

L'apprentissage autonome n'est pas seulement soumis au besoin d'efficacité de la technique de résolution. Il est également limité par la présence d'hyper-paramètres. Dans le cas des SVM, ces hyper-paramètres sont relativement peu nombreux mais un seul suffit à rendre une méthode dépendante d'une forme de supervision qui contredit soit le besoin d'apprentissage en ligne, soit l'objectif d'indépendance vis-à-vis d'une intervention humaine. Nous avons étudié ce problème par le biais des chemins de régularisation. Le chemin de régularisation permet de connaître toutes les solutions d'un problème au regard d'un compromis biais-variance. Pour les SVM, ce compromis est réglé par un des hyper-paramètres et nous utilisons donc le chemin de régularisation pour obtenir un réglage automatique de cet hyper-paramètre.

Contrairement à la première approche publiée pour le parcours du chemin de régularisation des SVM, nous ne partons pas de la solution prenant tous les exemples comme vecteurs supports. Au contraire, nous partons de la solution la plus parcimonieuse possible et le chemin est parcouru en direction de la solution la plus dense. Toutefois, dans un soucis d'efficacité et aussi en utilisant le fait qu'une bonne solution de SVM est parcimonieuse, nous arrêtons le parcours avant d'atteindre la solution dense. Pour déterminer le point d'arrêt, nous évaluons la qualité de la solution courante à chaque point d'arrêt à l'aide de la méthode du *leave-one-out*.

Cette approche permet de traiter des plus grandes bases de données que l'approche classique car elle utilise

la parcimonie. En revanche, pour ce qui est du réglage des hyper-paramètres, nous aurions besoin de pouvoir intégrer à cette méthode le réglage simultané de plusieurs hyper-paramètres.

Invariances et très grandes bases de données

Nous n'avons pas encore atteint le stade du *push-button SVM* mais nous montrons que toutes les limites des SVM ne sont pas insurmontables. En ce qui concerne la taille des bases d'apprentissage possibles à traiter en particulier, nous avons mis en œuvre le plus grand SVM à ce jour sur un seul processeur avec 80 millions de points en dimension 784. Le problème abordé est celui des invariances. En se replaçant dans l'optique de la détection de contexte, savoir traiter les invariances est un atout supplémentaire. Le cas présenté dans le chapitre 6 concerne la reconnaissance de l'écriture manuscrite mais ce n'est pas, et de loin, la seule application. En effet, ne serait-ce qu'au niveau du traitement des images, les cas d'invariances sont nombreux. On peut prendre pour exemple la base de données NORB [LeCun et al., 2004] qui propose des images d'objets sous différentes lumières et différentes positions. Au delà des images, les invariances existent aussi dans les signaux de toutes sortes. Une mélodie est reconnaissable si elle est transposée de quelques tons, une personne qui marche, quelque soit sa démarche, marche.

Pour ce qui est des invariances traitées par les SVM, nous donnons une approche générale pour intégrer la connaissance des variations à la structure même de la méthode de résolution et nous appliquons ce principe à SimpleSVM et à LASVM, avec succès. Pour peu que l'on sache formuler les invariances que l'on souhaite traiter, cette approche est généralisable directement à d'autres supports que les images.

Détection de ruptures

Nous en revenons maintenant à notre tâche première qui est la détection de contexte. Nous avons vu que parmi les briques nécessaires à la réalisation de notre architecture de traitement de données, seule celle concernant la détection de rupture dans les signaux a été mise en œuvre. L'idée de chercher à trouver les changements dans les signaux afin d'isoler des segments de données homogènes naît de l'observation de la trop grande diversité des signaux. Il semble déraisonnable de chercher à lister toutes les situations pour tous les utilisateurs. En particulier lorsqu'il s'agit d'utiliser des données physiologiques, la variabilité est très grande d'une personne à l'autre et difficilement caractérisable. Par conséquent nous avons choisi de travailler sur une approche non supervisée apte à découvrir autant de nouvelles classes que nécessaire. Le tri des séquences ainsi isolées est reporté à la couche suivante. L'intéret de ce procédé est de fournir à l'étape suivante des blocs de données homogènes qui seront considérés comme une seule entrée. L'approche présentée dans le chapitre 7 repose sur les tests statistiques et sur les méthodes à noyaux. Pour un instant donné, les données sont observées sur un court laps de temps avant et après, l'hypothèse est qu'il n'y a pas de changement de distribution entre les deux groupes.

Par l'utilisation des familles exponentielles, nous montrons comment faire un test statistique non paramétrique et nous donnons le lien qui existe entre ce résultat et le SVM à une classe. Forts de ce résultat, nous utilisons le SVM à une classe pour apprendre les données passées et tester les données futures. Ce test donne un score qui permet de prendre une décision, celle-ci étant basée sur plusieurs heuristiques dont celle du CUSUM. Cette démarche est testée sur des données réelles et montre son efficacité.

Au delà du résultat numérique, cette étude met en avant une complexité du problème qui relève de la subjectivité de la tâche. A partir de la vidéo des expériences, on demande à plusieurs personnes, dont la personne filmée, d'annoter les données, c'est-à-dire de donner les moments où, selon elles, il y a eu changement. On note qu'un

pourcentage non négligeable de divergences. Ce qui est intéressant dans la détection automatique c'est que sur nos deux expériences sur données réelles, on obtient pratiquement l'union des annotations manuelles. Procéder par méthode non supervisée semble donc apporter une notion d'objectivité. Toutefois ce résultat n'est obtenu que sur deux expériences et mériterait d'être étudié plus profondément.

Retour sur le déroulement de la thèse

Le projet de recherche de la thèse présenté ici est parti du constat qu'il manque une étape importante entre la miniaturisation des capteurs et l'utilisation des informations qu'ils sont aptes à capter. Cette étape, l'extraction des informations, relève du domaine de l'apprentissage automatique. Dans un premier temps nous avons eu pour objectif de « diviser pour régner » en proposant une approche en couche de la résolution de ce problème. Cette analyse du problème a rapidement exposé plusieurs verrous scientifiques en terme d'apprentissage autonome. Dès lors, nous avions deux options. La première était de continuer le développement des modules de notre architecture en comptant sur le perfectionnement futur des méthodes utilisées. La seconde était de prendre les choses en main et de contribuer directement à ce perfectionnement. Nous avons opté pour la seconde option. Ce virage dans les axes de recherche explique la diversité des travaux présentés, qui peut de prime abord paraître incongrue. Toutefois le fil conducteur, ténu, se retrouve dans chaque thème abordé : comment utiliser les méthodes d'apprentissage dans un contexte d'applications grand publique avec toutes les contraintes que cela sous-entend (beaucoup de données, temps réel, absence d'ingénieur qualifié pour régler le système à chaque dérive...) ?

Nous avons apporté, à divers niveaux, des éléments de réponses. Ces élément ne sont néanmoins pas encore suffisants au regard de notre objectif mais ouvrent plusieurs perspectives de recherche passionnantes. En particulier, nous prévoyons de nous atteler à la réalisation de deux tâches. Concernant les SVM, nous souhaitons travailler sur une méthode hybride entre les techniques de décomposition et les techniques de contraintes actives qui saurait s'adapter aux hyper-paramètres. Parallèlement à cela (et dans l'idéal, conjointement), nous étudierons les possibilités de régler tous les hyper-paramètres à l'aide de méthodes similaires aux chemins de régularisation. L'union de ces deux projets aboutirait au *push-button SVM*, le SVM utilisable sans expertise. Puisqu'il est reconnu que les SVM donnent des résultats précis dans la plupart des problèmes pour lesquels ils ont été utilisés, disposer d'un outil grand publique ouvre des perspectives très larges d'applications.

Bibliographie

Andreas Argyriou, Raphael Hauser, Charles A. Micchelli, and Massimiliano Ponti. A dc-programming algorithm for kernel selection. *ICML*, 2006.

W. Ark, D. Dryer, and D. Lu. The emotion mouse. In *Proceedings of HCI International. Munich, Germany*. 1999.

Francis Bach, David Heckerman, and Eric Horvitz. On the path to an ideal ROC curve : Considering cost asymmetry in learning classifiers. In Robert G. Cowell and Zoubin Ghahramani, editors, *AISTATS*, pages 9–16. Society for Artificial Intelligence and Statistics, 2005.

R. Banse and K.R. Scherer. Acoustic profiles in vocal emotion expression. *Journal of Personality and Social Psychology*, 70(3) :614–636, 1996.

Michèle Basseville and Igor V. Nikiforov. *Detection of Abrupt Changes - Theory and Application*. Prentice-Hall, 1993.

H. Bergson. Essai sur les données immédiates de la conscience. `http://www.uqac.uquebec.ca/zone30/Classiques_des_sciences_sociales/classiques/bergson_henri/essai_conscience_immediate/essai_conscience.pdf`, 1888.

S. Bidel, L. Lemoine, F. Piat, T. Artières, and P. Gallinari. Apprentissage de comportements utilisateurs de produits hypermédias. *RIA*, 17 :423–436, 2003.

Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. Working document : Fast kernel classifiers with online and active learning, May 2005a. http ://leon.bottou.com/publications/pdf/huller3.pdf.

Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6 :1579–1619, September 2005b.

Léon Bottou and Yann LeCun. Large scale online learning. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press.

Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2) :121–167, 1998. URL `citeseer.ist.psu.edu/article/burges98tutorial.html`.

J.T. Cacioppo and L.G. Tassinary. Inferring psychological significance from physiological signals. *Am Psychol*, 45 :16–28, 1990. doi : http://psychology.uchicago.edu/socpsych/faculty/jtcreprints/ct90.pdf.

S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy. Svm and kernel methods matlab toolbox. Perception Systèmes et Information, INSA de Rouen, Rouen, France, 2005.

Stéphane Canu and Alexander J. Smola. Kernel methods and the exponential family. *Neurocomputing*, 69(7-9) : 714–720, 2006.

C. Chang and C. Lin. LIBSVM : a library for support vector machines (version 2.3), 2001a. URL `citeseer.nj.nec.com/chang01libsvm.html`.

Chih-Chung Chang and Chih-Jen Lin. Training ν-support vector classifiers : Theory and algorithms. *Neural Computation*, (9) :2119–2147, 2001b.

O. Chapelle and B. Schölkopf. Incorporating invariances in nonlinear svms. In Dietterich T. G.and Becker S. and Ghahramani Z., editors, *Advances in Neural Information Processing Systems*, volume 14, pages 609–616, Cambridge, MA, USA, 2002. MIT Press.

P. H. Chen, C. J. Lin, and B. Schölkopf. A tutorial on v-support vector machines. *Applied Stochastic Models in Business and Industry*, 21(2) :111–136, 2005.

I.C. Christie and B.H. Friedman. Autonomic specificity of discrete emotion and dimensions of affective space : A multivariate approach. *International Journal of Psychophysiology*, 51 :143–153, 2004.

Antoine Cornuéjols, Laurent Miclet, and Yves Kodratoff. *Apprentissage Artificiel. Concepts et algorithmes*. 2002. ISBN 2-212-11020-0.

Roddy Cowie and Mark Schroder. Piecing together the emotion jigsaw. In *Lecture Notes in Computer Science*, volume 3361. Springer-Verlag Heidelberg, 2005.

Michael Crichton. *La Proie*. Robert Laffont, 2003.

M. Davy and S. Godsill. Detection of abrupt spectral changes using support vector machines. In *Proc. IEEE ICASSP-02*. 2002. URL `citeseer.ist.psu.edu/davy02detection.html`.

Fabien Delorme and Gaëlle Loosli. Un outil générique pour l'analyse automatique et la visualisation de productions d'apprenants. *TICE*, 2006.

F. Desobry, M. Davy, and C. Doncarli. An online kernel change detection algorithm. *IEEE Signal Processing*, 53 (5), May 2005.

B. Dorizzi, P. Lamadeleine, C. Guerrier, and J.L. Les Jardins. Biométrie : Techniques et usages. *Revue des sciences et techniques de l'ingenieur*, 2004.

R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley Interscience - 2e édition, 2001.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2) :407–499, 2004. URL `citeseer.ist.psu.edu/efron02least.html`.

Craig Fancourt and Jose C. Principe. On the use of neural networks in the generalized likelihood ratio test for detecting abrupt changes in signals. In *Intl. Joint Conf. on Neural Networks, pp. 243-248, at Como, Italy*. 2000.

M. Gertz and S. Wright. Object-oriented software for quadratic programming, 2001. URL `citeseer.ist.psu.edu/gertz01objectoriented.html`.

Thore Graepel and Ralf Herbrich. Invariant pattern recognition by semi-definite programming machines. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

Ulf Grenander. *General Pattern Theroy*. Oxford University Press, 1993.

Lacey Gunter and Ji Zhu. Computing the solution path for the regularized support vector regression. In *NIPS*, 2005.

Andreas Haag, Silke Goronzy, Peter Schaich, and Jason Williams. Emotion recognition using bio-sensors : First steps towards an automatic system. In *Lecture Notes in Computer Science*, volume 3068, pages 36–48. Springer-Verlag, 2004. doi : 10.1007/b98229.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5 :1391–1415, 2004.

J. Healey and R. Picard. Detecting stress during real-world driving tasks. *IEEE Trans. on Intelligent Transportation Systems*, 2005. To appear.

R. Herbrich. *Learning Kernel Classifiers*. MIT press, Cambridge, Massachusetts, 2002.

E. Hudlicka and M.D. McNeese. Special issue on the applications of affective computing in human-computer interaction. *International Journal of Human-Computer Studies*, 59(1-2) :1–255, 2003.

T. Joachims. Making large-scale SVM learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advanced in Kernel Methods - Support Vector Learning*, pages 169–184. MIT Press, 1999.

M. I. Jordan. Graphical models. *Statistical Science (Special Issue on Bayesian Statistics)*, 19 :140–155, 2004.

N. K. Karmarkar. A new polynomial–time algorithm for linear programming. *Combinatorica*, 4 :373–395, 1984.

D. Keysers, R. Paredes, H. Ney, and E. Vidal. Combination of tangent vectors and local representation for handwritten digit recognition. In *Lecture Notes in Computer Science*, volume LNCS 2396, pages 538–547. SPR2002, International Workshop on Statistical Pattern Recognition, Windsor, Ontario, Canada, springer-vertag edition, Aug 2002.

K. H. Kim, S. W. Bang, and S. R. Kim. Emotion recognition system using short-term monitoring of physiological signals. *Medical and biological engineering and computing*, 42, 2004.

Sunjung Kim and Elisabeth André. Composing affective music with a generate and sense approach. In *Proceedings of Flairs 2004 - Special Track on AI and Music*. AAAI Press, 2004.

J. Klein, Y. Moon, and R. W. Picard. This computer responds to user frustration : Theory, design, results, and implications. *Interacting with Computers*, 14 :119–140, 2002.

Kristof Van Laerhoven and Kofi Aidoo. Teaching context to applications. *Personal Ubiquitous Comput.*, 5(1) : 46–49, 2001. ISSN 1617-4909. doi : http://dx.doi.org/10.1007/s007790170029.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998. `http ://yann.lecun.com/exdb/mnist/`.

Y. LeCun, F.J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. *CVPR*, 2004.

J.-H. Lee and C.-J. Lin. Automatic model selection for support vector machines. Technical report, Dept. of Computer Science and Information Engineering, National Taiwan University, 2000.

Todd K. Leen. From data distributions to regularization in invariant learning. *Advances in Neural Information Processing Systems*, 7 :223–230, 1995.

Scott Lenser and Manuela Veloso. Non-parametric time series classification. In *Under review for ICRA'05*. 2005.

Xiangyang Li and Qiang Ji. Active affective state detection and user assistance with dynamic bayesian networks. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 35 :93– 105, 2005.

C. L. Lisetti and F. Nasoz. Using non-invasive wearable computers to recognize human emotions from physiological signals. *EURASIP Journal on Applied Signal Processing - Special Issue on Multimedia Human-Computer Interface*, 2004(11) :1672–1687, 2004.

Gwen Littlewort, Marian Stewart Bartlett, Ian Fasel, Joel Chenu, Takayuki Kanda, Hiroshi Ishiguro, and Javier R. Movellan. Towards social robots : Automatic evaluation of human-robot interaction by face detection and

expression classification. In *Advances in Neural Information Processing Systems 16*, pages 97–104. MIT Press, Cambridge, MA, 2004.

Gaëlle Loosli. Fast svm toolbox in Matlab based on SimpleSVM algorithm, 2004. `http ://asi.insa-rouen.fr/~gloosli/simpleSVM.html`.

Gaëlle Loosli and Stéphane Canu. Comments on the cvm. Technical report, LITIS Rouen, 2006a.

Gaëlle Loosli and Stéphane Canu. Stopping on the ν-svm regularization using LOO. Technical report, LITIS Rouen, 2006b.

Gaëlle Loosli, Stéphane Canu, and Alain Rakotomamonjy. Détection des activités quotidiennes à l'aide des séparateurs à vaste marge. *RJCIA2003 - 6eme Rencontres Nationales des Jeunes Chercheurs en Intelligence Artificielle*, pages 139–152, 2003.

Gaëlle Loosli, Stéphane Canu, S.V.N. Vishwanathan, Alexander J. Smola, and Monojit Chattopadhyay. Une boîte à outils rapide et simple pour les SVM. *CAp 2004 - Conférence d'Apprentissage*, pages 113–128, 2004.

Gaëlle Loosli, Stéphane Canu, SVN Vishwanathan, and Alexander J.Smola. Invariances in classification : an efficient svm implementation. *Applied Stochastic Models and Data Analysis*, 2005a.

Gaëlle Loosli, Stéphane Canu, S.V.N. Vishwanathan, Alexander J. Smola, and Monojit Chattopadhyay. Boîte à outils svm simple et rapide. *Revue d'intelligence artificielle*, 19, 2005b.

Gaëlle Loosli, Sang-Goog Lee, and Stéphane Canu. Context retrieval by rupture detection. *Conférence d'Apprentissage*, 2005c.

Gaëlle Loosli, Sang-Goog Lee, and Stéphane Canu. Context changes detection by one-class svms. *UM 2005 : Workshop on Machine Learning for User Modeling : Challenges*, 2005d.

Gaëlle Loosli, Sang-Goog Lee, and Stéphane Canu. Rupture detection for context aware applications. *CEUR Workshop*, 2005e. ISSN 1613-0073.

Gaëlle Loosli, Léon Bottou, and Stéphane Canu. Training invariant support vector machines using selective sampling. Technical report, LITIS - NEC Laboratories of America, 2006a.

Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Svm et apprentissage des très grandes bases de données. *CAp - Conférence d'apprentissage*, 2006b.

Gaëlle Loosli, Sang-Goog Lee, Vincent Guigue, Alain Rakotomamonjy, and Stéphane Canu. Perception d'états affectifs et apprentissage. *Revue d'intelligence artificielle, Edition spéciale Interaction Emotionnelle*, 20 No.4-5 :553–582, 2006c.

O. Mangasarian. Generalized support vector machines. *NIPS Workshop on Large Margin Classifiers*, 1998. URL `citeseer.ist.psu.edu/mangasarian98generalized.html`.

O. L. Mangasarian and David R. Musicant. Lagrangian support vector machines. *Journal of Machine Learning Research*, 1 :161–177, 2001.

Markos Markou and Sameer Singh. Novelty detection : a review, part 2 : neural network based approaches. *Signal Process.*, 83(12) :2499–2521, 2003. ISSN 0165-1684. doi : http://dx.doi.org/10.1016/j.sigpro.2003.07.019.

Charles A. Micchelli and Massimiliano Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6 :1099–1125, 2005.

Michel Minoux. *Programation mathématique. Théorie et algorithmes. Tome 1*. Dunod, 1983.

F. Nasoz, K. Alvarez, C. L. Lisetti, and N. Finkelstein. Emotion recognition from physiological signals for presence

technologies. *International Journal of Cognition, Technology, and Work – Special Issue on Presence*, 6(1), 2003.

Arkadi Nemirovski. Introduction to convex programming, interior point methods, and semi-definite programming. Machine Learning, Support Vector Machines, and Large-Scale Optimization Pascal Workshop, March 2005.

XuanLong Nguyen, Martin J. Wainwright, and Michael I. Jordan. Nonparametric decentralized detection using kernel methods. *IEEE Transactions on Signal Processing (accepted for publication)*, 2005.

Mila Nikolova. Local strong homogeneity of a regularized estimator. *SIAM Journal on Applied Mathematics*, 61 (2) :633–658, 2000.

E. Osuna, R. Freund, and F. Girosi. An improved training algorithm fo support vector machines. In J. Principe, L. Gile, N. Morgan, and E. Wilson, editors, *Neural Networks for Signal Processing VII - Proceedings of the 1997 IEEE Workshop*, pages 276–285, 1997.

M. Pantic and L.J.M. Rothkrantz. Toward an affect-sensitive multimodal human-computer interaction. *Proceedings of the IEEE*, 91(9), 2003. doi : 10.1109/JPROC.2003.817122.

R. W. Picard, S. Papert, W. Bender, B. Blumberg, C. Breazeal, D. Cavallo, T. Machover, M. Resnick, D. Roy, and C. Strohecker. Affective learning : A manifesto. *BT Technology Journal*, 22(4) :253–269, 2004. ISSN 1358-3948. doi : http://dx.doi.org/10.1023/B:BTTJ.0000047603.37042.33.

Rosalind W. Picard. Affective computing : challenges. *Int. J. Hum.-Comput. Stud.*, 59(1-2) :55–64, 2003.

Rosalind W. Picard, Elias Vyzas, and Jennifer Healey. Toward machine emotional intelligence : Analysis of affective physiological state. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(10) :1175–1191, 2001. ISSN 0162-8828. doi : http://dx.doi.org/10.1109/34.954607.

R.W. Picard. Response to sloman's review of affective computing. *AI Magazine*, 20(1) :134–137, 1999.

John Platt. Fast training of support vector machines using sequential minimal optimization. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advanced in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, 1999.

Norman Poh and Samy Bengio. Using chimeric users to construct fusion classifiers in biometric authentication tasks : An investigation. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2006.

Stephen Portnoy and Roger Koenker. The gaussian hare and the laplacian tortoise : computability of squared-error versus absolute-error estimators. *Statistical Sciences*, 1997.

A. Pozdnoukhov and S. Bengio. Tangent vector kernels for invariant image classification with svms. IDIAP-RR 75, IDIAP, Martigny, Switzerland, 2003. Submitted to International Conference on Pattern Recognition 2004.

Yuan Qi and Rosalind W. Picard. Context-sensitive bayesian classifiers and application to mouse pressure pattern classification. In *Proceedings of International Conference on Pattern Recognition*, 2002.

Liva Ralaivola and F. d'Alché Buc. Time series filtering, smoothing and learning using the kernel kalman filter. In *Proc. of IEEE Int. Joint Conference on Neural Networks, Montreal, Canada*, 2005.

P. Rani, N. Sarkar, A. Smith, C., and A. Adams, J. Affective communication for implicit human-machine interaction. *IEEE Transactions on Systems, Man, and Cybernetics*, 2005. Under Review.

Stephen Roberts, Evangelos Roussos, and Rizwan Choudrey. Hierarchy, priors and wavelets : structure and signal modelling using ica. *Signal Process.*, 84(2) :283–297, 2004. ISSN 0165-1684. doi : http://dx.doi.org/10.1016/j.sigpro.2003.10.012.

Chih-Jen Lin Rong-En Fan, Pai-Hsuen Chen. Working set selection using second order information for training

support vector machines. *Journal of Machine Learning Research*, 2005.

D. Roobaert. DirectSVM : A simple support vector machine perceptron. In *Neural Network for Signal Processing X - Proceedings of the 2000 IEEE Workshop*, pages 356–365. New York IEEE, 2000.

J. Scheirer, R. Fernandez, J. Klein, and R. W. Picard. Frustrating the user on purpose : A step toward building an affective computer. *Interacting with Computers*, 14(2) :93–118, 2002.

K. R. Scherer. Toward a dynamic theory of emotion : The component process model of affective states. `http://www.unige.ch/fapse/emotion/publications/pdf/tdte_1987.pdf`, 1987. Unpublished manuscript.

B. Schölkopf and A. Smola. *Leaning with Kernels*. MIT Press, 2001.

B Scholkopf, R Williamson, A Smola, and J Shawe-Taylor. Support vector method for novelty detection. In SA Solla, TK Leen, and KR Muller, editors, *NIPS*, pages 582–588. MIT Press, 2000.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, 2002.

B. Schölkopf, C. Burges, and V. Vapnik. Incorporating invariances in support vector learning machines. In J.C. Vorbrüggen C. von der Malsburg, W. von Seelen and B. Sendhoff, editors, *Artificial Neural Networks — ICANN'96*, volume 1112, pages 47–52, Berlin, 1996. Springer Lecture Notes in Computer Science.

P. Simard, Y. LeCun, and Denker J. efficient pattern recognition using a new transformation distance. In S. Hanson, J. Cowan, and L. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5. Morgan Kaufmann, 1993.

P. Y. Simard, Y. LeCun, J. S. Denker, and B. Victorri. Transformation invariance in pattern recognition – tangent distance and tangent propagation. *International Journal of Imaging Systems and Technology*, 11(3), 2000.

Patrice Simard, David Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. *ICDAR*, pages 958–962, 2003.

A Smola. Exponential families and kernels. Berder summer school, 2004. `http ://users.rsise.anu.edu.au/ smola/teaching/summer2004/`.

Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core vector machines : Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6 :363–392, 2005. ISSN 1533-7928.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6 :1453–1484, 2005.

R. J. Vanderbei. LOQO : An interior point code for quadratic programming. *Optimization Methods and Software*, 11 :451–484, 1999. URL `citeseer.ist.psu.edu/vanderbei98loqo.html`.

V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, 1982.

Vladimir Vapnik. *Statistical Learning Theory*. Wiley, 1998.

Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, N.Y, 1995.

S. V. N Vishwanathan, Alexander J. Smola, and M. Narasimha Murty. SimpleSVM. *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

G. Wahba. *Support Vector Machines, Reproducing Kernel Hilbert spaces and the randomized GACV*, pages 69–88. MIT Press, B. Scholkopf and C. Burges and A. Smola edition, 1999.

Hanna M. Wallach. Efficient training of conditional random fields. Master's thesis, Division of Informatics, University of Edinburgh, 2002.

Geoffrey I. Webb, Michael J. Pazzani, and Daniel Billsus. Machine learning for user modeling. *User Modeling and User-Adapted Interaction*, 11(1-2) :19–29, 2001. ISSN 0924-1868.

Jeffrey Wood. Invariant pattern recognition : A review. *Pattern Recognition*, 29 Issue 1 :1–19, 1996.

P Zimmermann, S Guttormsen, B Danuser, and P Gomez. Affective computinga rationale for measuring mood with mouse and keyboard. *Journal of Occupational Safety and Ergonomics (JOSE)*, 9(4) :539–551, 2003. http ://www.ciop.pl/7901.html.

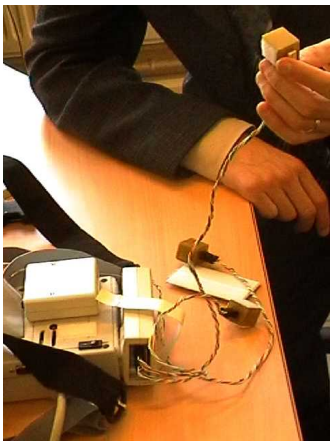
G. Zoutendijk. *Methods of Feasible Directions*. Elsevier, 1960.

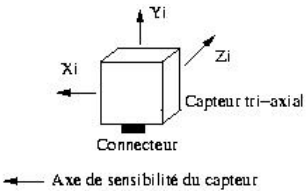
Annexe 1 - Capteurs

Accéléromètres

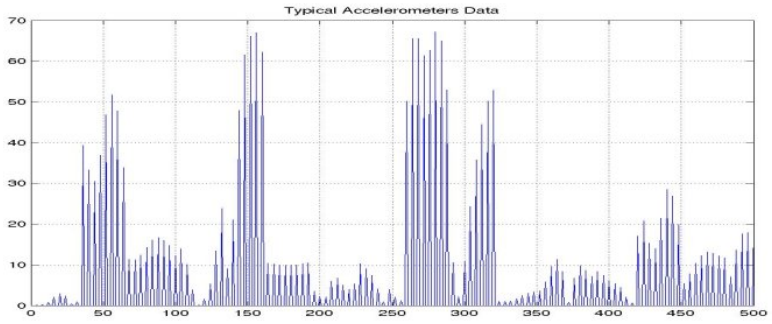
Le système utilisé pour l'acquisition des signaux accélérométriques nous a été prêté par l'ESIGELEC et Xavier Savatier. Le boitier d'acquisition se porte à la façon d'un sac à dos et les capteurs sont fixés sur une ceinture. Les données sont stockées sur une mémoire flash et transférées sur l'ordinateur à la fin de l'acquisition.

Accéléromètres





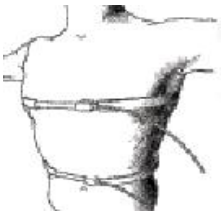
Accéléromètres tri-axiaux. Chaque accéléromètre donne l'accélération instantanée dans 3 directions. Ces capteurs sont positionnés autour des hanches et permettent de caractériser la démarche de l'utilisateur.



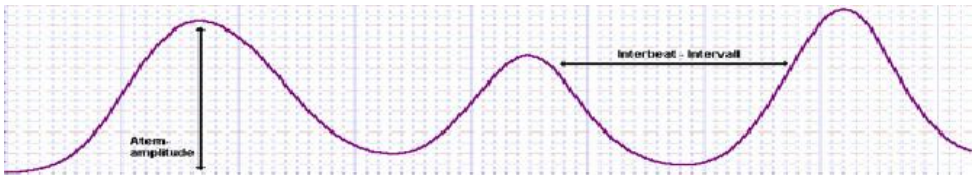
Procomp [3]

Le Procomp est un boitier d'acquisition de signaux biologique initiallement à vocation médicale. Nous avons disposé de cinq capteurs que nous présentons ici.

Respiration

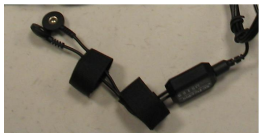


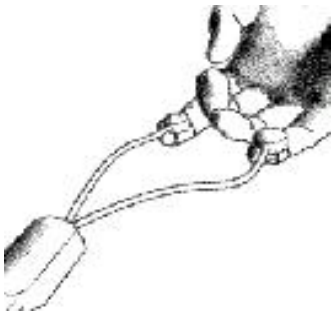
Capteur d'amplitude respiratoire. Ce capteur se présente sous la forme d'un tube de caoutchouc maintenu par une ceinture Velcro autour de la poitrine et d'un capteur électronique qui mesure l'étirement du tube en caoutchouc.



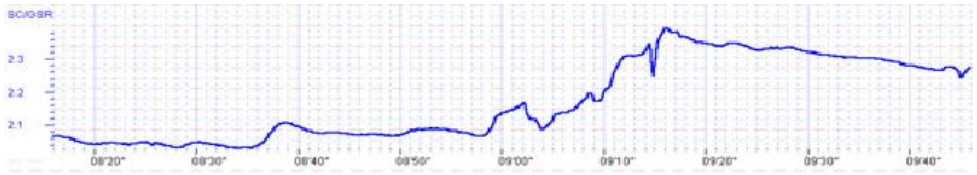
[3] images issues de la documentation du Procomp

Conductance de la peau

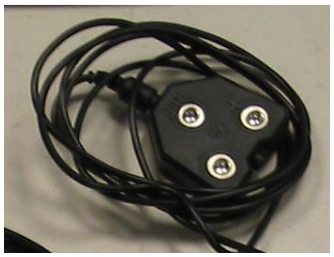


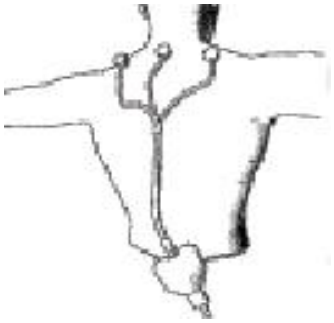


Capteur de conductance de la peau. Ce capteur mesure la capacitié de la peau à conduire l'électricité. Il est composé de deux électrodes (habituellement posées sur deux doigts d'une même main) entre lequelles circulent un faible courant. La peau conduit plus ou moins l'électricité en fonction de son humidité. Cette humidité est représentative du système nerveux sympatique. Plus une personne est stressée plus sa peau est conductive.

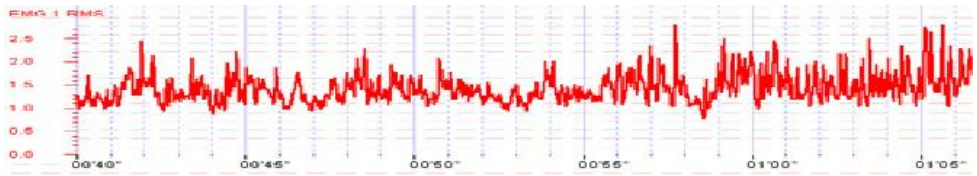


Electromyogramme

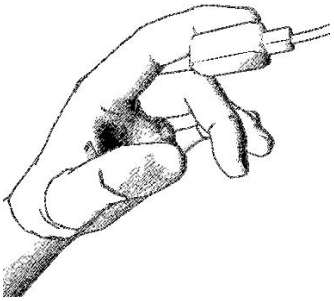




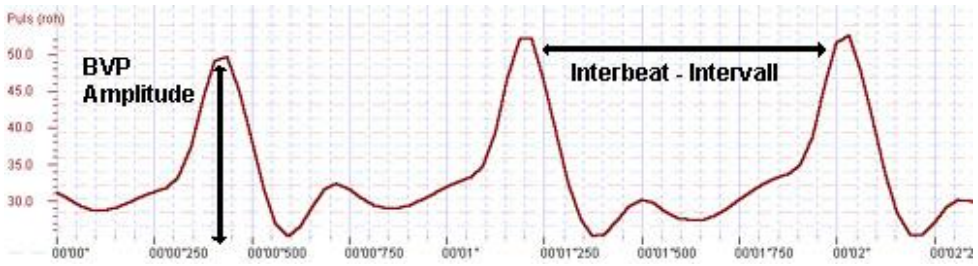
Capteur d'actvité électrique des muscles. Doté d'un pôle négatif et d'un pôle positif, le capteur enregistre les variations électriques créées par l'activité musculaire. L'amplitude du signal acquis est proportionnel à l'intensité de l'activité musculaire.



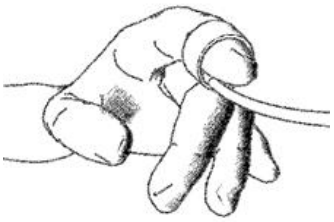
Pression sanguine



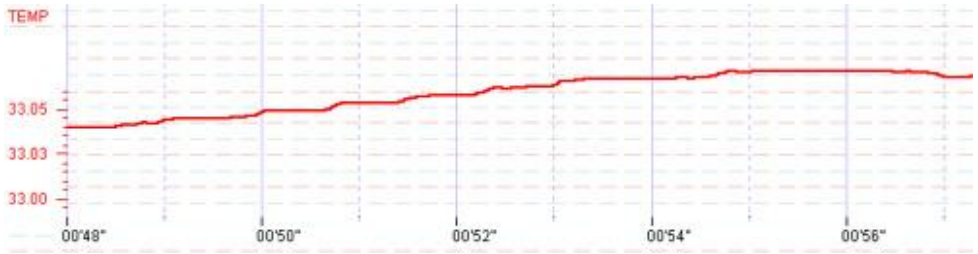
Capteur de pression sanguine. Ce capteur fonctionne avec l'envoie d'ondes infra-rouges suivi de la mesure de la quantité de lumière renvoyée par la peau. Cette quantité varie avec la quantité de sang présent dans la peau. A chaque battement de cœur, la quantité de sang dans la peau augmente ; comme le sang réflechit le rouge, la quantité de lumière réfléchie est plus grande. Le capteur peut être placé sur les doigts ou le lobe de l'oreille.



Température



Capteur de température périphérique. Ce capteur donne de petites variations lentes qui correspondent à l'évolutution de l'état de relaxation de la personne. La peau tend à se réchauffer quand l'individu se détend et vice-versa.



Annexe 2 - Bases de données

Bases synthétiques

Damier

Le jeu de données damier est un problème binaire séparable représentant un damier de 4 cases de côté.

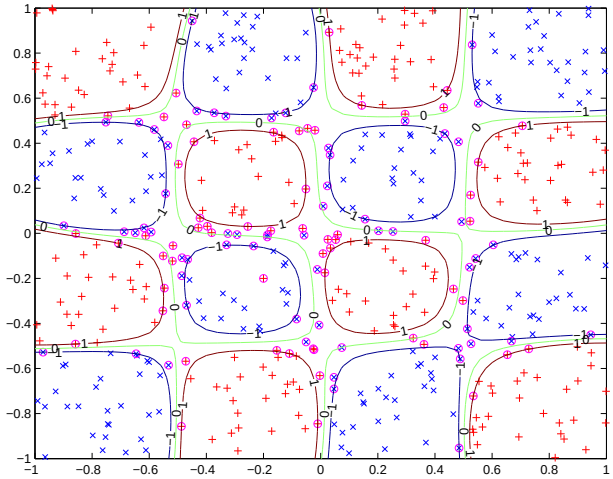


FIG. 5 : Jeux de données artificiels : damier

Pomme-Banane

Ce jeux de données, aussi appelé clown, est un problème binaire non séparable généré à l'aide de distributions gaussiennes.

Mixture

Le jeu est appelé « mixture » [4] et est un jeu de données non séparable avec une erreur de Bayes de 21%, utilisé dans Hastie et al. [2004] pour l'illustration du comportement du chemin de régularisation des SVM.

[4] disponible sur `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`

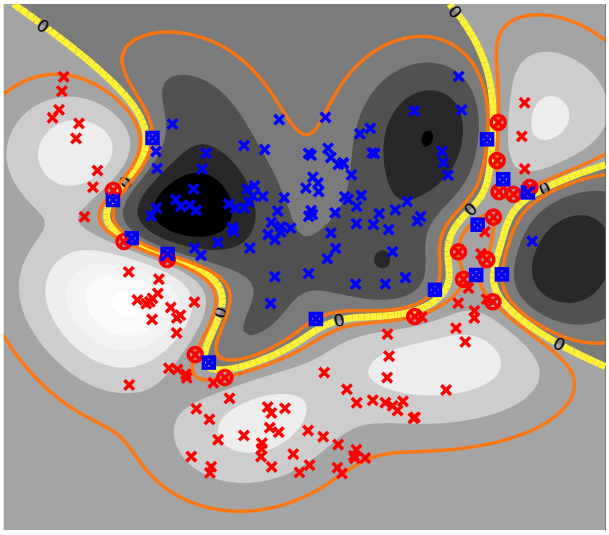


FIG. 6 : Jeux de données artificiels :pomme-banane.

Bases réelles

USPS

La base de données fournie par les services postaux américains contient près de 9000 imagettes représentant les chiffres de 0 à 9 extraits de codes postaux. Les données sont réparties entre des données d'apprentissage et des données de test. Chaque image est de dimension 16×16 pixels en niveaux de gris. L'erreur humaine en reconnaissance de la base de test est de 2.5%.

MNIST

MNIST est une base de données de reconnaissance d'écriture manuscrite contenant les chiffres de 0 à 9. La partie utilisée ici contient 60 000 images pour l'apprentissage et 10 000 pour le test. Les images sont de dimension 28×28 en niveaux de gris. 56 images sont *a priori* non identifiables par un humain.

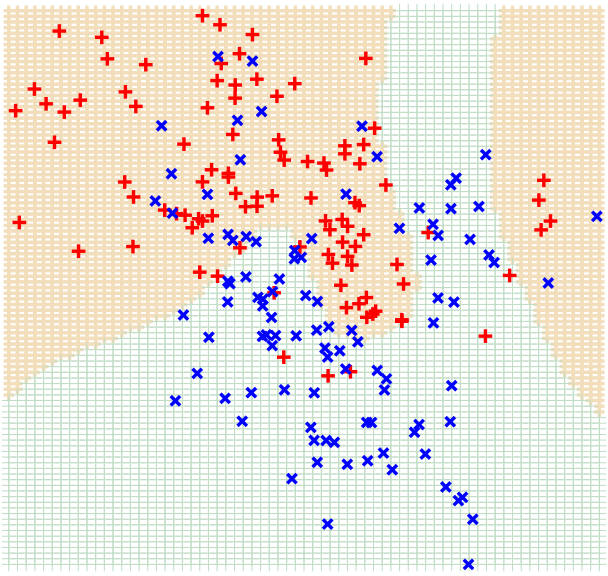


FIG. 7 : Jeux de données artificiels : mixture

Index

Symbols

Argyriou et al. [2006] 47
Ark et al. [1999] 10
Bach et al. [2005] 47
Banse and Scherer [1996] 9
Basseville and Nikiforov [1993] 85
Bergson [1888] 4
Bidel et al. [2003] 9
Bordes et al. [2005a] 8
Bordes et al. [2005b] 10, 26, 28
Bottou and LeCun [2004] 29
Burges [1998] 19
Cacioppo and Tassinary [1990] 9
Canu and Smola [2006] 86
Canu et al. [2005] 45, 59
Chang and Lin [2001a] 64
Chang and Lin [2001b] 65
Chapelle and Schölkopf [2002] 74
Chen et al. [2005] 65
Christie and Friedman [2004] 10
Cornuéjols et al. [2002] 6
Cowie and Schroder [2005] 5
Davy and Godsill [2002] 85
Desobry et al. [2005] 85
Dorizzi et al. [2004] 12
Duda et al. [2001] 6
Efron et al. [2004] 21
Fancourt and Principe [2000] 85
Gertz and Wright [2001] 25
Graepel and Herbrich [2004] 72, 75
Grenander [1993] 71
Gunter and Zhu [2005] 47
Haag et al. [2004] 9, 10
Hastie et al. [2001] 6, 8
Hastie et al. [2004] 47, 49, 52, 53, 55
Healey and Picard [2005] 9
Herbrich [2002] 7
Hudlicka and McNeese [2003] 4
Joachims [1999] 23, 62
Jordan [2004] 7
Karmarkar [1984] 25
Keysers et al. [2002] 80
Kim and André [2004] 9
Kim et al. [2004] 9, 10
Klein et al. [2002] 9
Laerhoven and Aidoo [2001] 9, 11
LeCun et al. [1998] 72
LeCun et al. [2004] 97
Lee and Lin [2000] 53
Leen [1995] 71
Lenser and Veloso [2005] 85
Li and Ji [2005] 9, 10
Lisetti and Nasoz [2004] 9
Littlewort et al. [2004] 10
Loosli and Canu [2006a] 59
Loosli and Canu [2006b] 47
Loosli et al. [2003] 16
Loosli et al. [2004] 10, 35
Loosli et al. [2005a] 71
Loosli et al. [2005b] 35
Loosli et al. [2005c] 85
Loosli et al. [2005d] 85
Loosli et al. [2005e] 85
Loosli et al. [2006a] 71
Loosli et al. [2006b] 71
Loosli et al. [2006c] 3
Loosli [2004] 59, 64
Mangasarian and Musicant [2001] 21
Mangasarian [1998] 21
Markou and Singh [2003] 85
Micchelli and Pontil [2005] 47
Minoux [1983] 38
Nasoz et al. [2003] 9, 10
Nemirovski [2005] 18
Nguyen et al. [2005] 85
Nikolova [2000] 21
Osuna et al. [1997] 24
Pantic and Rothkrantz [2003] 9, 10

Picard et al. [2001] 9, 10
Picard et al. [2004] 5, 11
Picard [1999] 4
Picard [2003] 12
Platt [1999] 25
Poh and Bengio [2006] 12
Portnoy and Koenker [1997] 21
Pozdnoukhov and Bengio [2003] 74
Qi and Picard [2002] 9, 10
Ralaivola and d'Alché Buc [2005] 8
Rani et al. [2005] 9
Roberts et al. [2004] 85
Rong-En Fan [2005] 25, 64
Roobaert [2000] 60
Schölkopf et al. [1996] 71, 74, 75, 80
Schölkopf and Smola [2002] 19, 20, 40, 61, 65, 84
Scheirer et al. [2002] 10
Scherer [1987] 9
Scholkopf et al. [2000] 7, 87
Schölkopf and Smola [2001] 7
Simard et al. [1993] 71, 74, 75
Simard et al. [2000] 75, 77
Simard et al. [2003] 84
Smola [2004] 87
Tsang et al. [2005] 26, 64, 65
Tsochantaridis et al. [2005] 45
Vanderbei [1999] 25
Vapnik [1982] 24
Vapnik [1995] 6, 19
Vapnik [1998] 8
Vishwanathan et al. [2003] 26, 39, 59, 64
Wahba [1999] 47, 53
Wallach [2002] 7
Webb et al. [2001] 10, 11
Wood [1996] 71, 75
Zimmermann et al. [2003] 9
Zoutendijk [1960] 23

A

accéléromètres 107
acquisition 107
affectif
    état affectif 3, 4
apprentissage 17, 18
    apprentissage en ligne 28, 44, 80, 91
    apprentissage non supervisé 4, 5, 7
    apprentissage statistique 4, 6
    apprentissage supervisé 4, 5, 7
    base d'apprentissage 19, 74, 111

B

boîte à outils 62

C

C-SVM 64
cache 61
capacité 19
capteur 107
    capteur de conductance 109
    capteur de pression sanguine 110
    capteur de respiration 108
    capteur de température périphérique 110
    capteurs biologiques 108
    capteur
        électromyogramme 109
champs de déformation 76
chemin de régularisation 47, 48
cholesky 60
compromis biais-variance 47
contexte 3
contraintes 37
contraintes actives 25, 38, 39, 59
coût 21, 38
    coût régulier 21
    coût singulier 21
    coût charnière 21
    régularisé 21
critère d'arrêt 61, 64
CVM 26, 64

D

décomposition 24
    chunking 24
    décomposition 24
    SMO 25
détection de changement 89
détection de rupture 86
direction admissible 23
domaine admissible 38
dual 22, 35, 41, 42, 73

E

effet d'échelle 65, 66
effet régularisant 66
émotion 3

espace
espace de hilbert . . . 20
espace pré-hilbertien . . . 20
état
état affectif . . . 4
perception d'état affectif . . . 9
état émotionnel . . . 4
perception d'émotion . . . 4
éthique . . . 12
étiquetage . . . 9

F

famille exponentielle . . . 86, 87
fonction de décision . . . 22

G

généralisation . . . 19
gradient . . . 38
gradient projeté . . . 38, 59
grandes dimensions . . . 48, 77

H

hyper-paramètres . . . 44, 64

I

implémentation . . . 59
instrumentation . . . 5
capteur . . . 5
invariances . . . 71

K

KKT . . . 23, 38, 61

L

lagrangien . . . 36, 41, 42, 73
LASVM . . . 28, 45
leave-one-out . . . 44, 52

M

maximum de vraissemblance . . . 87
MEB . . . 26

N

noyau . . . 20
ν-SVM . . . 42, 50, 64

O

OC-SVM . . . 40, 87
optimisation . . . 38, 50, 82
optimisation quadratique . . . 18, 22

P

parcimonie . . . 21, 23, 45, 48
point intérieur . . . 25
preuve de convergence . . . 39
primal . . . 22, 35, 40, 42, 73
produit scalaire . . . 20
projection . . . 38

Q

QR . . . 60

R

reconnaissance des formes . . . 6
régularisation . . . 21, 48
reprise à chaud . . . 43
résolution non optimale . . . 26
risque . . . 19
risque empirique . . . 19
RKHS . . . 20, 87

S

saut de dualité . . . 38, 61
segmentation de signal . . . 89
sélection de points . . . 28
sélection active . . . 29
sélection aléatoire . . . 28
sélection auto-active . . . 29
sélection par gradient . . . 28
seuil . . . 87
seuil mobile . . . 87
sherman-morrison . . . 61
shrinking . . . 23
simpleSVM . . . 59
simpleSVM invariant . . . 74
SMO . . . 28
SVM binaire . . . 22
système linéaire . . . 38, 60

T

test statistique . . . 86
théorème de représentation . . . 20

V

validation croisée . . . 44
VC-dimension . . . 19
vecteurs tangents . . . 75