وزارة التعليـم العـالـــي والبحـث العلــمـي

BADJI MOKHTAR - ANNABA UNIVERSITY

UNIVERSITE BADJI MOKHTAR - ANNABA

جـامعـة بـاجـي مخـتـار - عـنـابـة

Année : 2014/2015

Faculté des Sciences de l'Ingénieur

Département de l'Informatique

THESE

Présentée en vue de l'obtention du diplôme de doctorat en Sciences.

Intitulée :

Accélération de l'Apprentissage des SVMs pour le traitement de bases de données de grandes tailles : Application à la reconnaissance des mots arabes isolés.

Par : M^me^ CHAIBI Sonia

Pr MEROUANI Hayet Farida : Professeur à l'Université d'Annaba - Président

Pr LASKRI Mohamed Tayeb : Professeur à l'Université d'Annaba - Rapporteur

Pr AISSANI-MOKHTARI Aicha : Professeur à l'Université USTHB Alger - Examinateur

Pr SI-MOHAMED Malik : Professeur à l'Université de Tizi Ouzou - Examinateur

Pr BOUDOUR Rachid : Professeur à l'Université d'Annaba - Examinateur

Pr AMIRAT Abdelkrim : Professeur à l'Université de Souk Ahras - Examinateur

Université Badji Mokhtar – Annaba

A *mes Parents.*

A *mes enfants .*

REMERCIEMENTS

Au nom de dieu clément et miséricordieux le plus grand merci lui revient de nous avoir guidé vers le droit chemin, de nous avoir aidé tout au long de nos études, et de nous avoir aidé à réaliser ce travail.

Je tiens à exprimer mes respectueux remerciements et ma profonde reconnaissance envers mon Directeur de Thèse Monsieur le Professeur Laskri Mohamed Tayeb qui m'a orienté et conseillé tout au long de mon travail, qu'il soit vivement remercié.

Sonia

Dédicace

Je dédie cette thèse à :

-Mes parents bien aimés

-Mon mari :Aissa

-Mes enfants : Sourour, Hamza et Nour

-Mon frère :Nacer.

Ma sœur Amel ,son mari Abdelwaheb et à leurs enfants :Souad,Mohamed,Nardjes et Mayar
Ma sœur Houda,son mari Djamel et leurs enfants : MohdSalah et Nesrine

- Mes sœurs : Sameh et Affef

Sonia

Résumé

La révolution numérique a facilité considérablement la capture, la saisie et le stockage des données à un coût pratiquement nul. Comme conséquence de ceci, des énormes quantités de données de grandes dimensions (Images, Video …ect) sont stockées dans des bases de données de façon continuelle. La classification de telles bases de données est un problème délicat qui apparaît dans de nombreuses disciplines de l'informatique telle que la reconnaissance de l'écriture arabe manuscrite. Raison pour laquelle le développement de méthodes de classification automatiques pour ces bases de données est primordial.

Les Séparateurs à Vaste Marge est une méthode de classification qui montre de bonnes performances dans divers problèmes tels que le traitement d'image, la catégorisation de textes ou le diagnostique médical et ce même sur des ensembles de données de tailles importantes.

Néanmoins, la réalisation d'un programme d'apprentissage par SVMs se ramène à résoudre un problème de programmation quadratique (QP) dont la taille mémoire et le temps nécessaires pour sa résolution sont linéairement liés à la taille des exemples utilisés. Quand on a affaire à des bases de données de grandes tailles, la mise en œuvre des SVMs devient encore plus couteuse en espace mémoire et en temps d'exécution.

Pour surmonter cette insuffisance des SVMs, une nouvelle proposition utilisant le partitionnement K-moyenne au sein d'un espace réduit du vecteur caractéristique des données est développée dans thèse. Nos résultats montrent qu'il est désormais possible d'utiliser les SVMs pour classifier des bases d'une taille d'exemples importante. La reconnaissance des noms manuscrits des Wilayas Algériennes en est un exemple type traité avec succès dans cette thèse.

Mots clés : Séparateur à vaste Marge (SVM), Regroupement K-moyenne, Réduction de dimension, Analyse Discriminante Linéaire, Reconnaissance des Mots Arabes Manuscrits.

Abstract

The digital revolution has greatly facilitated the capture, the capture and the storage of data at virtually no cost. As a consequence of this, huge amounts of high-dimensional data (Images, Video ... ect) are stored in databases continuously. The classification of such high-dimensional data is a difficult problem that appears in many disciplines such as pattern recognition. Why the development of automatic methods for the classification of these databases is essential.

Support vector Machines is a classification method that shows good performance in various problems such as image processing, text categorization or medical diagnostic and even on data sets of very large dimensions.

However, the implementation of an SVMs learning program is reduced to solving a quadratic programming problem (QP) whose memory size et time required for its resolution are linearly related to the size examples et in the case of large databases the implementation of SVMs becomes more expensive in space memory et in time.

To overcome this disadvantage of SVMs, a new proposal using the K-means within a reduced future space is presented in this work. Our results show that it is now possible to use SVMs to solve problems with a large samples size. The recognition of handwritten names of Algerian Wilaya is a typical example successfully treated in this paper.

Keywords: Support Vector Machines,K-means Clustering, dimensionality reduction, Linear discriminant Analyse, Arabic Handwriting Words Recognition.

الملخـــــــــص

الثورة الرقمية قد سهلت كثيرا من الالتقاط، والقبض وتخزين البيانات تقريبا بدون تكلفة. ونتيجة لذلك، يتم تخزين كميات هائلة من البيانات الكبيرة (صور، فيديو ... إلخ) في قواعد البيانات بشكل مستمر. تصنيف قواعد البيانات هذه هي مشكلة صعبة وتظهر في العديد من التخصصات مثل التعرف الآلي على خط اليد العربية. لهذا أصبح تطوير أساليب التصنيف الآلي لقواعد البيانات أمر ضروري.

طريقة تصنيف الفاصل ذو الهامش الواسع تظهر أداء جيد في مختلف المشاكل مثل معالجة الصور، وتصنيف النص أو في التشخيص الطبي حتى على أحجام كبيرة من مجموعات البيانات.

غير أن تنفيذ برنامج التعلم SVMs يعود إلى حل مشكلة البرمجة التربيعية (QP) و الذاكرة الحجم والوقت اللازم لقرارها ترتبط خطيا بحجم الأمثلة المستخدمة. عند التعامل مع قواعد البيانات كبيرة الحجم، وتنفيذ SVMs يصبح أكثر تكلفة في مساحة الذاكرة والوقت الوحدة المعالجة المركزية.

للتغلب على هذا النقص في SVMs اقتراح جديد باستخدام متوسط K-التقسيم ضمن فضاء مميزة ناقلات البيانات وضعت في هذه الدراسة. نتائجنا تظهر أن أصبح من الممكن الآن استخدام SVMs لتصنيف حجم العينة كبير. التعرف على أسماء مكتوبة بخط اليد مثل التعرف على الولايات الجزائرية هو مثال نموذجي عولج بنجاح في هذه الدراسة.

كلمات البحث: فاصل كبير هامش (SVMs)، تجمع K-المتوسط، وتخفيض الحجم، الخطي التحليل المميز، التعرف على الكلمات المخطوطات العربية.

Table de matières

REMERCIEMENTS 3

..... 4

Dédicace 4

Résumé 5

Abstract 6

الملخـــــص 7

Liste des Figures 11

Liste des Tableaux 12

Introduction générale 13

Chapitre 1 : 16

L'Apprentissage Automatiques et la Classification des données. 16

1.1 Généralités 17

1.2 La classification en théorie 18

1.2.1 Formulation 18

1.2.2 Modèle génératif des données 19

1.2.3 Fonction d'erreur et risque 19

1.2.4 Machine d'apprentissage 20

1.2.5 Risque empirique 21

1.3 Analyse statistique de l'apprentissage 21

1.3.1 Insuffisance de MRE 21

1.3.2 Conditions de validité de MRE 23

1.3.3 Dimension VC 30

1.3.4 Minimisation du risque structurel (MRS) 33

1.3.5 Régularisation du risque 33

1.4 La classification en pratique 34

1.4.1 Ensemble de données 34

1.4.2 Entraînement ou apprentissage 35

1.4.3 Evaluation du modèle (test) 35

1.4.4 Exploitation 36

1.5 Quelques Méthodes de Classification 36

1.5.1 Approches statistiques 36

1.5.2 Méthodes paramétriques 37

1.5.3 Méthodes non paramétriques 39

1.5.4 Les Réseaux de Neurones 41

1.5.5 Classification par Modèle de Markov Caché 44
Conclusion 46
Chapitre II : 47
Les Machines à Vecteurs de support. 47
1.1 Introduction 48
1.2 Risque structurel 49
2.3 Espace augmenté 52
2.4 Formulation 54
2.4.1 Hyper-plans de séparation 55
2.4.2 Hyper-plans à marge optimale 56
2.4.3 Hyper-plans à marge molle 57
2.4.4 Le SVM non-linéaire 59
2.4.5 Conditions de Karush-Kuhn-Tucker 60
2.4.6 Calcul du biais **b** 61
2.5 Discriminant de Fisher non-linéaire (KFD) 62
2.6 Analyse en Composantes Principales non-linéaire 64
2.7 Classification mono-classe 66
2.8 Algorithmes d'apprentissage du SVMs 68
2.8.1 Méthode de chunking 69
2.8.2 Méthode de décomposition successive 69
2.8.3 Méthode de minimisation séquentielle: SMO 70
2.8.4 Algorithme de Joachims 70
2.9 Classification de données multi-classes 73
Conclusion 82
Chapitre III : 83
Techniques pour la Réduction de Dimension 83
3.1 Introduction 84
3.2 Propriétés des espaces à grandes e dimension 86
3.3 Techniques linéaires de Réduction de dimension. 89
3.3.1 Analyse en Composantes Principales (ACP) 89
3.3.2 Analyse en Composantes Indépendantes (ACI) 93
3.3.3 Analyse Linéaire Discriminante (LDA) 96
3.4 Techniques de réduction non- linéaires. 100
3.4.1 Isomap 101
3.4.2 Plongement localement linéaire 103

Chapitre IV : **104**
Contribution et Expérimentations. **104**
4.1 Introduction **105**
4 .2 Mise en pratique des Séparateurs à Vaste Marge: **106**
4 .3 Proposition d'une nouvelle approche de sélection de vecteurs supports: **108**
4.4 Résultats expérimentaux de l'approche de sélection par K-moyenne: **111**
4.4.1 Résultats Expérimentation de la base Ripley : 111
4.4.2 Résultats expérimentaux de la base Banana : 114
4.4.3 Autres bases de données: 118
4.5 Sélection de données au sein d'un espace LDA : **119**
4.5.1 Calcule du modéle LDA : 120
4.5.1 Résultats expérimentaux de la combinaison LDA/K-moyenne : 121
4.6 Application à la reconnaissance des Mots Arabes manuscrits : **124**
4.6.1 Bases de données des noms des Wilayas Algériennes: 126
4.6.2 Prétraitement de l'image : 126
4.6.3 Les Caractéristiques Structurelles : 127
4.6.4 Les caractéristiques statistiques : 127
4.6.7 Résultats expérimentaux de la sélection proposée : 127
Conclusion **128**
Conclusion et perspectives **129**
Références **133**

Liste des Figures

FIGURE1.1: APPRENTISSAGE AUTOMATIQUE SUPERVISE. $f(x)$ EST UNE ESTIMATION DE $f(x)$ 17
FIGURE 1.2- UNE FONCTION AYANT UN ,COMMETTANT DE NOMBREUSES. 22
FIGURE 1.3 - L'AXE DES ABSCISSES REPRESENTE LES FONCTIONS DE **F**. 25
FIGURE 1.4- EXEMPLE DE CONFIGURATION DE 3 POINTS SEPARABLES DE TOUTES LES MANIERES POSSIBLES ……..31
FIGURE1.5- EVOLUTION DU TERME DE CONFIANCE ……..32
FIGURE 1.6 - MINIMISATION DU RISQUE STRUCTUREL..... 33
FIGURE 1.7 - ENTRAINEMENT D'UNE MACHINE D'APPRENTISSAGE..... 35
FIGURE 1.8: EXEMPLE D'UN MELANGE DE TROIS DENSITES GAUSSIENNES BIDIMENSIONNELLES..... 39
FIGURE 1.9 : DECISION SELON LE PLUS PROCHE VOISIN. 40
FIGURE1.10: DECISION SELON LES K PLUS PROCHES VOISINS. 41
FIGURE 1.11 : EXEMPLE DE RESEAU DE NEURONES. FIGURE 1.12 : FONCTIONNEMENT D'UN NEURONE. 42
FIGURE 1.13: QUELQUES FONCTIONS DE SEUILS. 43
FIGURE 1.14 : STRUCTURE D'UN PERCEPTRON A TROIS COUCHES. 44
FIGURE 1.15: EXEMPLE DE MODELE DE MARKOV CACHE. 45

FIGURE 2. 1 : ARBRE DE CLASSIFICATION DES METHODES D'APPRENTISSAGE A BASE DE NOYAUX 48
FIGURE 2. 2 : TRANSFORMATION DES DONNEES AVEC LES NOYAUX DE MERCER 53
FIGURE 2. 3 : VARIATION DE LA CORRELATION POUR LES NOYAUX KMOD,. 59
FIGURE2. 5 : DISCRIMINANT DE FISHER: 63
FIGURE 2. 6 : DECOUVERTE DE STRUCTURES PAR PROJECTION DES DONNEES 65
FIGURE 2. 7 : SURFACE DE L'HYPER-SPHERE ENGLOBANTE DANS L'ESPACE D'ENTREE POUR DEUX TYPES DE NOYAUX. 67
FIGURE 2. 8 : PROBLEME A TROIS CLASSES 74
FIGURE 2. 9: ARCHITECTURE DU SYSTEME EN STRATEGIE UN-CONTRE-TOUS..... 75
FIGURE 2. 10 : ARCHITECTURE DU SYSTEME EN STRATEGIE UN-CONTRE-UN 78

FIGURE3. 1: PROPRIETE NON INTUITIVE DE LA BORDURE D'UN HYPERCUBE EN GRANDE E DIMENSION 88
FIGURE3. 2 : PROJECTION ORTHOGONALE DU POINT xi SUR UNE DROITE PASSANT PAR LE CENTRE DE GRAVITE μ . . 91
FIGURE3. 3 : EXEMPLE DE REDUCTION DE DIMENSIONS PAR LDA ET PAR ACP. 98

FIGURE4. 1: FRONTIERE DE DECISION D'UNE SEULE ITERATION POUR K =30 112
FIGURE4. 2: BORDURE DE DECISION OBTENUE PAR L'APPROCHE PROPOSEE. 113
FIGURE4. 3: L'ENSEMBLE D'APPRENTISSAGE DE LA TROISIEME ITERATION..... 116
FIGURE4. 4: L'ENSEMBLE DE DONNEES SELECTIONNE POUR LA PROCHAINE ITERATION EN GRAS..... 116
FIGURE4. 5: SUPPRESSION DES ZONES MIXTES 116
FIGURE4. 6: COMPARAISON DES FRONTIERES DE DECISIONS 117
FIGURE4. 7 : SCHEMA GENERAL DES ETAPES IMPORTANTES DE LA CLASSIFICATION DES MOTS ARABES PAR SVMS. . 125
FIGURE4. 8 : DONNEES EN PRESERVANT UNIQUEMENT LES CARACTERISTIQUES ESSENTIELS DE LA FORME..... 126

Liste des Tableaux

TABLEAU2.1: QUELQUES NOYAUX CLASSIQUES 54
TABLEAU2. 2:FONCTIONS D'ACTIVATION POUR LA RECONSTRUCTION 78

TABLEAU3. 1: NOMBRE DE COINS D'UN HYPERCUBE EN FONCTION DU NOMBRE DE DIMENSIONS M 86

TABLEAU4. 1: RESULTATS OBTENUS DE L'EXECUTION D'UNE ITERATION 111
TABLEAU4. 2: RESULTATS DE L'APPROCHE PROPOSEE. 112
TABLEAU4. 3: RESULTATS OBTENUS POUR RIPLEY. 113
TABLEAU4. 4: RESULTATS DES ITERATIONS DE LA PROCEDURE DE SELECTION POUR BANANA. 115
TABLEAU4. 5: COMPARAISON DES RESULTATS OBTENUS. 117
TABLEAU4. 6: CARACTERISTIQUES DES BASES DE DONNEES UCI CHOISIES. 118
TABLEAU4. 7 : RESULTATS OBTENUS. 118
TABLEAU4. 9 : RESULTATS OBTENUS POUR SVMGUIDE3. 122
TABLEAU4. 10: RESULTATS OBTENUS POUR WAVE. 123
TABLEAU4. 11: RESULTATS OBTENUS POUR LETTER. 123
TABLEAU4. 12:COMPARAISON DES RESULTATS OBTENUS. 128

Introduction générale

La Reconnaissance des Formes (RF) dont le but consiste à associer une étiquette (une classe) à une donnée qui peut se présenter sous forme d'une image , d'un signal… etc., est un axe fort important du domaine de l'intelligence artificielle, qui trouve application dans beaucoup de systèmes comme les interfaces visuelles, l'analyse de données, la bioinformatique, le multimédia,… etc. Plusieurs méthodes ont été développées dans ce domaine, en particulier les Réseaux de Neurones avec le perceptron multicouche (MLP). Ces classificateurs ont fait 1'objet de nombreuses recherches et ont donné des résultats impressionnants. Et depuis longtemps, les perceptrons multicouches ont été utilisés avec succès pour résoudre de nombreux problèmes de classification en RF.

Mais, de plus en plus, avec la diversité des applications de la Reconnaissance de Formes, plusieurs problèmes non linéaires faisant intervenir la classification sont rendus complexes. Ainsi, depuis quelques années, pour contourner la complexité des fonctions de décision construites pour les problèmes non linéaires, les recherches ont donné naissance à de nouvelles méthodes de classification basées sur les noyaux *(kemel methods)* qui s'avèrent plus robustes et plus simples que d'autres méthodes telles que les couches cachées introduites dans les Réseaux de Neurones.

La méthode des noyaux est une technique récente en reconnaissance de formes. Elle consiste à projeter les données qui sont initialement non séparables dans un autre espace de dimension plus élevée où elles peuvent le devenir. Les classificateurs utilisant la méthode des noyaux, contrairement aux classificateurs traditionnels, construisent la fonction de décision dans un nouvel espace autre que 1'espace des caractéristiques d'entrée; ce qui permet de réduire la complexité de la fonction de décision tout en gardant une bonne performance du classificateur. Les machines à vecteurs de supports (SVMs), issues des travaux de Vapnik **[Vapnik,1995],** utilisent cette technique des noyaux qui permet de traiter linéairement dans le nouvel espace les problèmes préalablement non linéaires.

Les SVMs constituent une famille de classificateurs basés sur le principe du risque structurel qui leur confère un fort pouvoir de généralisation. La minimisation du risque

structurel permet d'éviter le phénomène de sur-apprentissage des données. Ainsi, le risque structurel permet de garantir une estimation moins biaisée du risque réel. Bien qu'elles réduisent considérablement le nombre de paramètres de réglages par rapport à certaines méthodes classiques comme les Réseaux de Neurones, l'apprentissage des SVMs reste fortement dépendant de la taille de l'espace des données à traiter.

Les algorithmes d'apprentissage SVMs se déroulent à partir de la seule connaissance des valeurs de noyau $k(a_i,a_j)$ entre les données d'apprentissage. Ces valeurs sont habituellement mémorisées dans une matrice carrée que l'on appelle "matrice de Gram" et qui est de taille N×N (où N est le nombre de données d'apprentissage). Optimiser le critère d'apprentissage dual des SVMs par programmation quadratique requiert un espace mémoire $O(N^2)$ et une complexité calculatoire $O(N^3)$.

L'objectif du travail réalisé dans cette thèse est le développement d'une heuristique permettant l'accélération de l'apprentissage des SVMs sur des bases de données de tailles importantes tout en gardant la même précision des approches analytiques. L'idée est de réduire efficacement l'ensemble de données pour obtenir un ensemble de vecteurs supports suffisant pour l'apprentissage tout en préservant la même précision et en diminuant le facteur temps. Cet objectif est atteint par la réalisation d'un regroupement K-moyenne des données au sein d'un espace réduit du vecteur caractéristique généré par L'Analyse Discriminante Linéaire(LDA).

Les performances des algorithmes développés sont ensuite évaluées sur différentes bases de données : les résultats sont équivalents aux algorithmes usuels en ce qui concerne les taux de précision. Nos principaux apports sont d'une part l'amélioration du temps de réponse du modèle et d'autre part la capacité à traiter de très grandes quantités de données sur du matériel standard.

Cette thèse est constituée de quatre chapitres :

Dans le premier chapitre, nous introduisons les concepts fondamentaux de l'apprentissage supervisé. Nous validerons ensuite cette approche en effectuant une analyse statistique de l'apprentissage, principalement basée sur la théorie de Vapnik-Chervonenkis. Ensuite, nous présenterons les principales étapes de la mise en pratique de

la méthodologie Machine Learning (ML) et à la fin du chapitre, nous exposons les détails des approches de classification les plus répandues.

Dans le deuxième chapitre, on présente une introduction à la théorie de l'apprentissage statistique dont dérive le principe du risque structurel. En second lieu, nous présentons la théorie des machines à vecteurs de support après un bref rappel historique des travaux qui ont le plus influencé la théorie. En outre, on y trouve une analyse des principaux algorithmes d'apprentissage des SVMs.

Le troisième chapitre, présente l'état de l'art des techniques de la réduction de dimensionnalité. Nous présentons en détail les techniques de réduction par extraction de caractéristiques. Une revue de quelques méthodes linéaires et non linéaires est effectuée.

Le dernier chapitre est consacré à l'explication de notre proposition de sélection des vecteurs supports basée sur le regroupement K-moyenne. Ensuite nous montrons l'importance de la sélection par K-moyenne au sein d'un espace réduit et son effet immédiat en qualité de sélection et en réduction du temps et d'espace de traitement. Les performances de la méthode proposée sont mises en évidence à travers une comparaison avec les méthodes SVMs de la littérature du domaine et ceci en utilisant un nombre important de bases de données allant de la plus simple à la plus compliquée.

Une conclusion générale est présentée en fin de mémoire pour résumer le travail effectué dans cette thèse ; elle est suivie des perspectives possibles du travail.

Chapitre 1 :

L'Apprentissage Automatique et la Classification des données.

1.1 Généralités

La résolution de problèmes par la construction de machines capables d'apprendre à partir d'expériences caractérise l'approche fondamentale du Machine Learning (ML). Parmi les techniques du ML on trouve entre autres les méthodes supervisées et non-supervisées. Dans le cas des méthodes supervisées, on cherche à estimer (à apprendre) une fonction f(x) à partir d'exemples. La figure 1.1 illustre un tel processus. Les méthodes de classification font partie des techniques de ML supervisées.

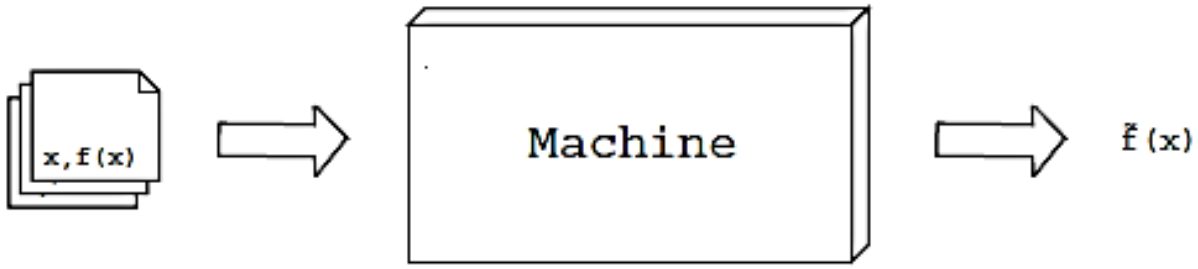


Figure1.1: Apprentissage automatique supervisé. $\tilde{\boldsymbol{f}}(\boldsymbol{x})$ est une estimation de $\boldsymbol{f}(\boldsymbol{x})$

Dans ce cadre, un exemple[1] consiste en un objet ou sa représentation (un document, une image, les caractéristiques d'une fleur, ...) accompagné de la catégorie à laquelle il appartient. La fonction f(x) que l'on va chercher à estimer, détermine la relation entre les objets et leur catégorie. Un classificateur sera donc caractérisé par la fonction f(x) qu'il implémente. On dira qu'un classificateur a un grand pouvoir de généralisation s'il fournit de bons résultats sur des données différentes de celles qui ont été utilisées pour l'apprentissage.

Dans les techniques non-supervisées, les objets sont présentés sans leur catégorie.

Un exemple type de méthodes non-supervisées est le clustering dont le but est de créer des groupes (clusters) d'objets présentant des caractéristiques semblables.

[1] On emploiera aussi le terme "donnée"

1.2 La classification en théorie

Dans l'introduction, nous avons donné une idée intuitive de ce qu'est la classification automatique. Dans cette section, nous nous employons à la formaliser en définissant sa forme mathématique et en introduisant le contexte statistique requis par l'approche Machine Learning Supervisé.

1.2.1 Formulation

La tâche qu'un classificateur doit effectuer peut être exprimée par une fonction que l'on appelle fonction de décision: **[Duda et al, 2000]**

$$f: x \rightarrow y$$

où x est l'ensemble des objets à classer (aussi appelé espace d'entrée)

y est l'ensemble des catégories (aussi appelé espace d'arrivée)

Dans la suite du document, nous nous limitons à la classification binaire.

Dans ce cas, l'ensemble y correspond à $\{+1, -1\}$. La plupart du temps on interprétera $+1$ et -1 respectivement comme l'appartenance et la non-appartenance à une catégorie déterminée. Une grande partie de la littérature se focalise sur le cas binaire parce que les classifications faisant intervenir plus de 2 catégories (cas multi-classes) peuvent toujours être ré-exprimées sous forme de classifications binaires. En effet, si l'on considère une classification multi-classes, nous pouvons effectuer, pour chaque catégorie, une classification binaire indépendante et regarder ensuite pour quelle catégorie l'appartenance s'est manifestée. Notons aussi que l'on peut assigner plusieurs catégories à un objet. Ce dernier cas, souvent appelé classification multi-label **[Duda et al, 2000]**, ne sera cependant pas abordé dans cette thèse.

Jusqu'à présent nous n'avons fait que définir le domaine et l'image de la fonction à estimer. Bien entendu, on ne peut pas accepter n'importe quelle fonction de cette forme. Nous devons en quelque sorte imposer que la fonction représente bien la relation entre les objets et leur catégorie. Pour ce faire nous avons besoin de modéliser le processus selon

lequel les données sont générées et d'introduire une fonction de coût indiquant à quel point notre fonction f s'écarte de ce processus.

1.2.2 Modèle génératif des données

Suivant la théorie de Vapnik **[Vapnik, 1995]**, nous supposons que les données sont générées selon une distribution de probabilités inconnue $P(x, y)$. De plus nous supposons les données **i**ndépendantes et **i**dentiquement **d**istribuées (iid).

Notons deux choses:

- Cette hypothèse permet aussi d'englober le cas multi-labels. En effet $P(x, y)$ permet d'associer une distribution de catégories à chaque objet x.

- Dans ce modèle on fait "comme si" une source unique produisait les données (x, y) de manière iid. On conçoit aisément que ce modèle délibérément simple ne soit pas adapté à tous les cas. Si la distribution change au cours du temps (non-stationnaire) ou si les données ne sont pas complètement générées de façon indépendante, il conviendra d'utiliser un modèle plus fin. Dans le cadre de cette thèse nous nous positionnons volontairement dans le cas où ce modèle est suffisant.

1.2.3 Fonction d'erreur et risque

Si nous disposons de n exemples bien classés $(x_1, y_1) \dots (x_n, y_1)$, une première approche pour déterminer les performances d'un classificateur est de comparer ses prédictions avec les classes y_i attendues. A cette fin, on introduit une fonction d'erreur.

Définition 1.1 (fonction d'erreur) : Soit le triplet $(x, y, f(x)) \in x \times y \times y$ où x est un objet, y sa catégorie et f (x) la prédiction du classificateur. Toute fonction $c: x \times y \times y \rightarrow +[0, \infty[$ telle que $c(x, y, y) = 0$ est appelée fonction d'erreur.

Par la suite nous utilisons une fonction d'erreur simple et bien adaptée à la classification binaire :

$$E(x, y, f(x)) = \frac{1}{2}|f(x) - y| \qquad (1.1)$$

Nous introduisons à présent la notion de risque (en anglais : functional risk) qui représente l'erreur moyenne commise sur toute la distribution $P(x, y)$ par la fonction $f(x)$:

$$R[f] = \int_{x\,x\,y} \frac{1}{2} |f(x) - y| \, dP(x, y) \qquad (1.2)$$

Nous sommes à présent en mesure d'introduire un critère selon lequel la fonction f devra être optimale :

$$f^{opt} = \arg_f \min(R[f]) \qquad (1.3)$$

En d'autres termes, la fonction f^{opt} devra être telle que l'erreur moyenne sur toute la distribution soit minimale.

Le critère que nous venons de formuler est malheureusement inutilisable tel quel. En effet pour calculer le risque nous devrions disposer d'une estimation de la distribution $P(x, y)$, ce qui n'est pas le cas.

On prend souvent le classificateur aléatoire comme point de référence. Il s'agit du classificateur qui assigne aléatoirement une catégorie aux objets. Un tel classificateur possède une erreur moyenne de 50%, ce qui est le plus mauvais score admissible. En effet, si un classificateur donne un score plus bas, il suffit d'inverser sa fonction de décision pour obtenir le score complémentaire. Un classificateur dont la seule restriction est de faire mieux que le classificateur aléatoire est appelé classificateur faible (weak classifier).

1.2.4 Machine d'apprentissage

On désigne par machine d'apprentissage, une machine dont la tâche est d'apprendre une fonction à travers des exemples. Une machine d'apprentissage est donc définie par la classe de fonctions F qu'elle peut implémenter. Dans notre cas, ces fonctions sont des fonctions de décision. Nous notons F une famille de fonctions telle que chacun de ses membres est caractérisé par une valuation unique des paramètres γ_i. A titre d'exemple considérons la famille qui représente l'ensemble des fonctions de décision d'un classificateur linéaire élémentaire: le perceptron.

$$F_{\gamma}(x) = \text{sign}(\sum_{i=1}^{n} \gamma_i x_i + \gamma_0)$$

On note $f_{\acute{\gamma}}$ la fonction de F_{γ} instanciée avec le paramètre $\acute{\gamma}$.

1.2.5 Risque empirique

Si on dispose d'une machine d'apprentissage et d'un ensemble de t_i exemples, on peut exprimer le risque empirique d'une fonction $f_{\acute{\gamma}}$:

$$R_{emp}[f_{\acute{\gamma}}] = \sum_{i=1}^{n} \frac{1}{2} |f_{\acute{\gamma}}(x_i) - y_i| \qquad (1.4)$$

De manière similaire à la section précédente, on peut dériver un critère de sélection de la fonction optimale f^n :

$$f^n = f_{\acute{\gamma}} \text{ telle que } \gamma^* = \arg_{\acute{\gamma}} \min(R_{emp}|f_{\acute{\gamma}}|) \qquad (1.5)$$

Ce critère est appelé **m**inimisation du **r**isque **e**mpirique (MRE).

1.3 Analyse statistique de l'apprentissage

1.3.1 Insuffisance de MRE

On peut légitimement se demander si le critère de minimisation du risque empirique mène toujours à un classificateur possédant un bon pouvoir de généralisation. En d'autres termes: la minimisation du risque empirique implique-t-elle la minimisation du risque ? Nous allons tenter de répondre à cette question au moyen d'un exemple simple.

Supposons que nous disposions d'un ensemble de n documents annotés (avec leur catégorie), représentés dans $\mathbb{R}^2$. Un document est donc représenté par deux réels, par exemple le nombre d'occurences des termes "paix" et "guerre". La figure 1.2 représente la situation; les exemples positifs et négatifs sont respectivement représentés par des x et des o. On voit que la fonction ne fait aucune erreur d'assignation sur les documents utilisés lors de l'apprentissage (la partie gauche). Par contre, sur de nouveaux documents (partie droite), on observe que la moitié d'entre eux sont mal classés (même performance que le classificateur aléatoire).

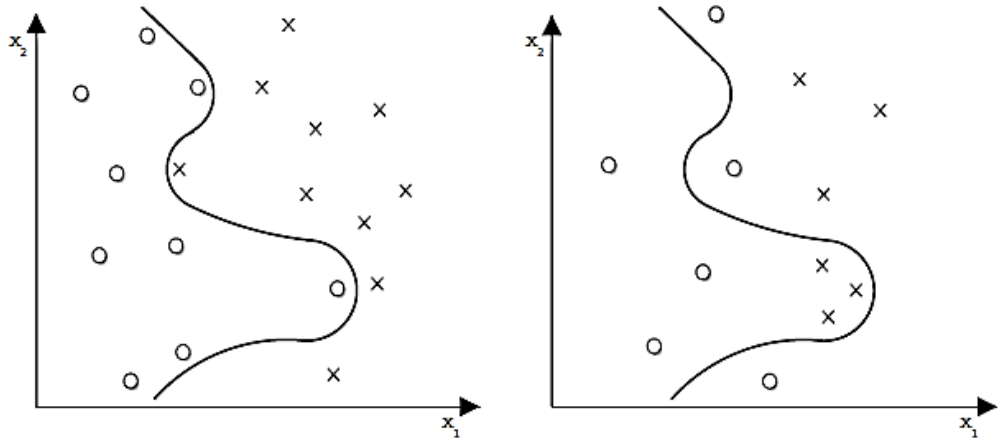


Figure 1.2- Une fonction ayant un risque empirique nul (à gauche), commettant de nombreuses erreurs sur un jeu de données inconnu (à droite).

Nous venons de voir que si on n'impose aucune restriction sur la classe F on peut toujours trouver une fonction f qui se comporte bien vis-à-vis du risque empirique, sans pour autant assurer une bonne généralisation sur de nouveaux documents. La restriction des fonctions implémentables nous confronte à un dilemme que les statisticiens appellent biais-variance **[Schôlkopf et al, 2002**].

Dans la terminologie Machine Learning on l'appelle surgénéralisation - overfitting. Concrètement cela dit deux choses:

- D'une part, en limitant fortement la taille de l'ensemble F, on tend à expliquer la relation entre les objets et leur classe de manière trop grossière (surgénéralisation). Dans ce cas le risque empirique sera élevé mais le modèle ne "collera" pas aux exemples de trop près.

- D'autre part, si on admet un grande nombre de fonctions, la relation sera modélisée de manière complexe et le bruit associé aux mesures (acquisition d'image, de caractères, ...) risque également d'être appris (overfitting). On parle souvent d'apprentissage par cœur parce que le classificateur aura un risque empirique très faible mais ses performances sur d'autres jeux de données seront mauvaises.

1.3.2 Conditions de validité de MRE

Dans cette sous-section, nous effectuons une analyse statistique du type Vapnik-Chervonenkis de MRE. Par souci de clarté, nous préférons présenter une analyse quelque peu simplifiée plutôt qu'un traitement mathématique tout à fait rigoureux. Les résultats obtenus ne seront par conséquent pas les plus raffinés, cependant les concepts dégagés fourniront de bons éléments de réponse à notre question.

1.3.2.1 *Loi des grands nombres*

Avant de chercher à restreindre F, observons les résultats obtenus par MRE lorsque le nombre de données n est arbitrairement grand.

Supposons que l'on présente n objets à un classificateur binaire de fonction de décision f fixée. On associe à chaque objet l'erreur $\xi_i = \frac{1}{2}|f(x_i) - y_i|$ valant 0 ou 1 suivant que le classificateur prédit bien ou mal. On peut considérer les $Ç_i$ comme des épreuves de Bernoulli indépendantes, relatives à la variable aléatoire $\xi = \frac{1}{2}|f(x) - y|$.

L'inégalité de Chernoff [**Chernoff**, **1952**] nous donne une borne probabiliste sur l'écart existant entre la moyenne empirique $\frac{1}{n}\sum_{n=1}^{n}\xi_i$ et l'espérance de ξ :

$$P\left\{\left|\frac{1}{n}\sum_{i=1}^{n}\xi_i - E(\xi)\right| \geq \epsilon\right\} \leq 2\exp(-2\,\epsilon^2) \qquad (1.6)$$

que l'on peut réécrire en utilisant les équations (1.2) et (1.4)

$$P\{|R_{emp}[f] - R[f]| \geq \epsilon\} \leq 2\exp(-2n\,\epsilon^2) \qquad (1.7)$$

En d'autres termes, la convergence en probabilité :

$$|R_{emp}[f] - R[f]| \xrightarrow{P} 0$$

est exponentiellement rapide en le nombre de données n. La probabilité dans (1.6) et (1.7) porte sur le fait d'obtenir un ensemble de n données, tel que l'écart entre les risques pour une fonction fixée f, soit supérieur à ϵ ($R_{emp}[f]$ est donc la variable aléatoire). Bien

que le résultat ci-dessus soit essentiel, certaines considérations nous engagent à plus de prudence :

Le résultat obtenu est de nature probabiliste; il peut donc exister des cas où l'écart entre les risques est élevé. Par ailleurs, nous n'avons pas encore opéré de restriction sur F; l'espace de recherche est potentiellement très grand. Il se pourrait donc qu'en minimisant R_{emp} on tombe, par malchance, sur une fonction pour laquelle il existe un large écart entre R et R_{emp} .

Lorsque l'on entraîne une machine avec le critère MRE, on cherche activement une fonction f telle que la moyenne empirique des ξ_i soit aussi petite que possible. Nous ne sommes donc plus dans le cas d'épreuves de Bernouilli puisque que la fonction f n'a pas encore été fixée et que nos n données seront utilisées à cette fin.

1.3.2.2 *Consistance et convergence uniforme*

La notion de consistance introduite par Vapnik **[Vapnik., 1995]** formalise les conditions requises pour la validité de MRE.

Définition 1.2 (consistance) La méthode MRE est dite consistante sur un ensemble de fonctions F si et seulement si

$$R_{emp}(f^n) \xrightarrow{P} R(f^{opt}) \qquad (1.8)$$

$$R(f^n) \xrightarrow{P} R(f^{opt}) \qquad (1.9)$$

où f^{opt} , f^n sont respectivement les minimisants sur f de R et R_{emp} .

La consistance impose que lorsque le nombre de données n tend vers l'infini

- la valeur du R_{emp} pour la fonction sélectionnée par MRE converge vers la plus petite valeur atteignable pour R.
- la solution de R_{emp} soit également une solution optimale de R. Remarquons que cela n'impose pas l'unicité des solutions f^{opt} et f^n:

Sans prendre davantage de restrictions, il s'avère que la méthode MRE n'est pas consistante. On peut expliquer cette inconsistance par le caractère local[2] de l'inégalité de Chernoff [**Chernoff, 1952**]. En effet[2], nous savons, que pour chaque fonction de F, les risques convergent en probabilité, cependant il se pourrait que les vitesses de convergence varient d'une fonction à l'autre sur F .Nous illustrons ces propos par la figure 1.3. On voit que pour un n arbitrairement grand, le minimisant f^n choisi par MRE peut ne pas correspondre au minimisant f^{opt} de R. Ce résultat un peu déroutant s'explique par le fait que même s'il existe un $n_\in$ arbitrairement grand, à partir duquel les risques de toutes les fonctions ont convergé à $\in$ près, on peut toujours examiner la convergence de plus près (avec $\acute{\in} < \in$) et être confronté aux différences de vitesse potentielles. Pour pallier ce problème, Vapnik a montré qu'il faut exiger une convergence qui soit uniforme[3] sur toutes les fonctions et ce quelle que soit la précision de $\in$.

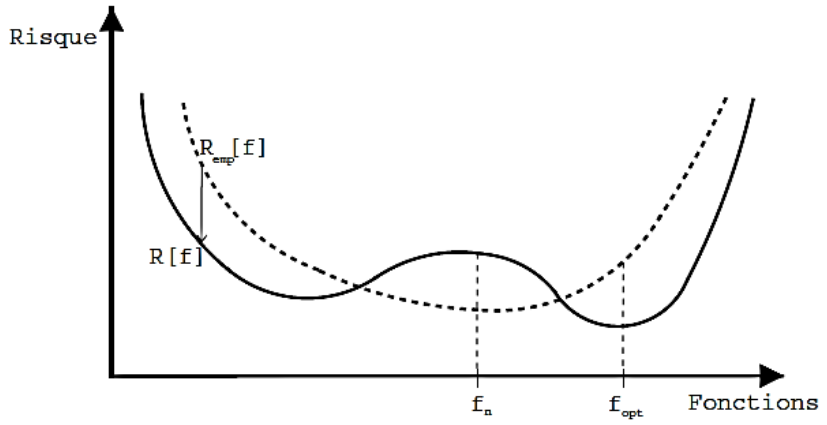


Figure 1.3 - L'axe des abscisses représente les fonctions de **F** (il s'agit d'une représentation imagée car rien n'indique qu'un tel ensemble est dense) et celui des ordonnées, la valeur des risques. Les courbes pleines et pointillées correspondent respectivement à **R** et à $\mathbf{R_{emp}}$. La flèche verticale indique que pour chaque $\mathbf{f} \in \mathbf{F}$, les risques convergent selon Chernoff.

La détermination de la consistance doit faire l'objet d'une analyse sur le pire cas. En faisant le rapprochement avec la loi des grands nombres (section 1.3.2.3), la consistance est conditionnée par la fonction pour laquelle les risques convergent le moins vite.

[2] - qui n'adresse qu'une fonction à la fois

[3]- En d'autres mots, il faut que les vitesses de convergence soient identiques pour toutes les fonctions

Nous introduisons à présent un théorème fondamental qui fait le lien entre la notion de consistance et la convergence uniforme des risques basée sur le pire cas.

Théorème 1 (Vapnik & Chervonenkis) La convergence uniforme

$$\lim_{n\to\infty} P\{\sup_{f\in F}(R|f| - R_{emp}|f|) > \in\} = 0 \quad (1.10)$$

$\forall \in > 0$ est une condition suffisante et nécessaire pour la consistance de MRE.

1.3.2.3 *Approche de la convergence uniforme*

Le théorème précédent nous fournit donc l'équivalence entre la consistance et la convergence uniforme. Cependant, l'application d'une telle condition pose problème: le membre de gauche de (1.10) reste très difficile à évaluer à cause de notre manque d'informations concernant le risque R. Pour commencer, nous allons considérer un cas simple dans lequel le nombre de fonctions implémentables est fini. La démarche opérée servira de base pour le cas, quelque peu plus subtile, où la cardinalité de F est infinie.

- **La cardinalité de F est finie**

Dans ce cas, nous allons exprimer (1.10) sous la forme d'une union finie d'évènements que nous pouvons ensuite borner en utilisant l'inégalité de Chernoff.

Soit $C_{\in}^{i} = \{(x_1, y_1), \dots \dots \dots, (x_n, y_n) \, |R[f_i] - R[f_i]| > \in\}$,

un ensemble de n exemples pour lequel les risques de f_i divergent de plus de $\in$. Prenons un cas simple à deux fonctions: $F = \{f_1, f_2\}$. Dans ce cas, (1.10) correspond à la probabilité que les évènements $C_{\in}^{1}$ ou $C_{\in}^{2}$; se réalisent :

$$P\{\sup_{f\in F}(R[f] - R_{emp}[f]) > \in\} = P\{C_{\in}^{1} \cup C_{\in}^{2}\}$$

que nous pouvons développer

$$P\{C_{\in}^{1} \cup C_{\in}^{2}\} = P\{C_{\in}^{1}\} + P\{C_{\in}^{2}\} - P\{C_{\in}^{1} \cap C_{\in}^{2}\} \quad (1.11)$$

$$\leq P\{C_{\in}^{1}\} + P\{C_{\in}^{2}\} \quad (1.12)$$

Par extension, on peut appliquer ce raisonnement à un ensemble F de cardinalité m.

$$P\{\sup_{f\in F}(R[f] - R_{emp}[f]) > \epsilon\} = P\{C_{i=1}^{m} C_{\epsilon}^{i}\} \quad (1.13)$$

$$= \sum_{i=1}^{m} P\{C_{\epsilon}^{i}\} - \xi \quad (1.14)$$

$$\leq \sum_{i=1}^{m} P\{C_{\epsilon}^{i}\} \quad (1.15)$$

Il est maintenant possible de borner chaque terme de la somme de (1.15) par l'inégalité de Chernoff et d'obtenir une borne:

$$P\{\sup_{f\in F}(R[f] - R_{emp}[f]) > \epsilon\} = 2m\, exp(-2n\,\epsilon^2)\} \quad (1.16)$$

Nous pouvons faire remarquer deux choses:

- Nous nous trouvons dans le cas de l'égalité si les ensembles sont disjoints, ce qui est rarement le cas en pratique.
- Cette borne dépend directement de la cardinalité m de l'ensemble de fonctions considéré. Par conséquent elle ne sera d'aucune utilité pour des machines pouvant implémenter un nombre infini de fonctions.

➢ **La cardinalité de *F* est infinie**

La plupart du temps les machines d'apprentissage peuvent implémenter une infinité de fonctions en faisant varier les paramètres r_i qui caractérisent les différents membres de la classe F (par exemple l'ensemble des fonctions de décision linéaires). Pour obtenir une borne utilisable, nous devons trouver un moyen de nous ramener dans un cas fini dans lequel la cardinalité de l'ensemble F n'intervient pas en tant que telle.

Nous commençons par émettre une observation qui sera à la base d'un lemme permettant la réduction au cas fini. Lorsque l'on parle de R_{emp}, nous regardons les valeurs des fonctions de F sur un ensemble fini de points $x_1 \ldots x_n$. Or les fonctions de décision sont des fonctions binaires et nous savons que sur un ensemble de n points il existe au plus 2^n fonctions binaires possédant des valuations distinctes.

Rappelons que la difficulté d'adresser directement (1.10) vient du fait que le risque fait intervenir toute la distribution de données qui nous est inconnue et potentiellement infinie.

Nous introduisons à présent un lemme permettant d'approcher (1.10) au moyen d'un évènement exprimé en termes de risques empiriques faisant donc référence à un nombre fini d'exemples, et par conséquent, de fonctions (après réduction). Le lemme requiert d'avoir un nombre pair d'exemples pour exprimer une symétrie. Nous prendrons dès lors les conventions suivantes:

2n: le nombre d'exemples considérés

$\acute{R}_{emp}$: le risque empirique sur les n premiers exemples

$\acute{\acute{R}}_{emp}$: le risque empirique sur les n derniers exemples

Lemme 1 (Symétrie) $\forall n \in \geq 2$, nous avons

$$P\{\sup_{f\in F}(R[f] - R_{emp}[f]) > \in\} \leq P\{\sup_{f\in F}(\acute{R}_{emp}[f] - \acute{\acute{R}}_{emp}[f]) > \frac{\in}{2}\} \qquad (1.17)$$

La démonstration du lemme se trouve dans **[Vapnik, 1995].** Intuitivement, cela exprime que si les risques empiriques des deux ensembles de données iid sont proches, leurs valeurs doivent être proches de celle du risque R. Notre stratégie en vue de l'obtention d'une borne va être identique à celle du cas fini. Nous allons exprimer la partie droite de (1.17) au moyen d'une somme dont chaque terme peut être borné au moyen de l'inégalité de Chernoff.

Avant cela, nous introduisons le coefficient d'éclatement (en anglais: shattering coefficient) qui relève du concept de capacité. De manière informelle, une capacité se rapporte à une famille de fonctions et donne une mesure de la complexité avec laquelle cette famille peut séparer des points. Dans le cas présent, ce coefficient va nous servir à énumérer les fonctions de F distinctes sur 2n exemples. Par la suite nous introduirons un autre terme de capacité offrant d'intéressantes propriétés quant à la borne obtenue.

Définition 1.3 (coefficient d'éclatement) Le coefficient d'éclatement, noté $S(F, n)$ est le nombre maximum d'assignations distinctes de labels que les fonctions de F peuvent réaliser sur un ensemble arbitraire de n points.

Remarquons que le coefficient indique qu'il doit exister au moins une configuration de n points permettant d'atteindre ce résultat. Cela ne garantit donc pas que l'on peut séparer n'importe quel ensemble de n points de toutes ces manières. Par ailleurs, pour être plus précis, nous devons restreindre les fonctions de F sur les points que nous considérons et non pas sur la pire configuration de points.

Nous reformulons les ensembles

$$C = \left\{(x_1, y_1) \ldots\ldots (x_{2n}, y_{2n}) \left| (\acute{R}_{emp}[f] - \acute{\acute{R}}_{emp}[f]) \right| > \frac{\epsilon}{2} \right\}$$

et le corollaire de Chernoff associé :

$$P\left\{\frac{1}{n}\sum_{i=1}^{n} \xi_i - \frac{1}{n}\sum_{i=n+1}^{2n} \xi_i \geq \epsilon\right\} \leq 2 \exp -(-\frac{n\,\epsilon^2}{2}$$

A présent nous avons en main tous les outils nécessaires pour borner (1.10)

$$P\{\sup_{f\in F}(R[f] - R_{emp}[f] > \epsilon\} \leq 2P\left\{\sup_{f\in F}(\acute{R}_{emp}[f] - \acute{\acute{R}}_{emp}[f] > \frac{\epsilon}{2}\right\} \quad (1.18)$$

$$\leq 2P\left\{\cup_{i=1}^{S(F,2n)} C^i_{\epsilon/2}\right\} \quad (1.19)$$

$$\leq 2 \sum_{i=1}^{S(F,2n)} \{PC^i_{\epsilon/2}\} \quad (1.20)$$

$$\leq 2\, S(F, 2n) \exp(\frac{-2\epsilon^2}{8}) \quad (1.21)$$

Une manière plus pratique d'exprimer cette borne est de construire un intervalle de confiance. Dès lors, nous obtenons une borne qui tient avec une probabilité 1 - δ :

$$R[f] \leq R[f] + \sqrt{\frac{8}{n}(\ln(S(F, 2n)) + \ln\frac{4}{\delta})} \qquad (1.22)$$

Remarques

- Cette formulation est très intéressante, essentiellement parce qu'elle permet d'approcher le risque sans avoir besoin de disposer de la distribution des données $P(x, y)$. Nous devons malgré tout rester prudents car il s'agit d'une borne. En minimisant cette borne, nous ne minimisons pas forcément le risque R.

- Le second terme du membre de droite est appelé terme de confiance. Sa présence nous incite à prendre un nouvel élément en compte dans la recherche de la fonction de décision. En effet, en plus du critère MRE nous allons veiller à ce que la famille de fonctions dans laquelle possède un terme de confiance intéressant.

- Ce type de borne nous permet davantage de dégager des concepts qu'une utilisation immédiate dans la pratique. Il faut aussi se rappeler que tout le raisonnement est fondé sur une limite pour laquelle le nombre de données tend vers l'infini or en pratique, nous disposons d'un nombre limité de données. Certains auteurs ont cependant développé des bornes du même type adaptées à des algorithmes d'apprentissage spécifiques. **([Smola et Scholkopf, 1998], [Vapnik, 1995])**

- Le terme de confiance est fortement lié à la notion de capacité. La plupart du temps on minimise indirectement le terme de confiance au travers de la capacité[4]. Dans cette formulation, il est possible d'utiliser un autre terme de capacité. On cherche dès lors à utiliser une mesure de capacité adéquate au problème considéré et facile à manipuler.

1.3.3 La Dimension de Vapnik-Chervonenkis

Dans le chapitre 2, nous présentons une méthode de classification basée sur un hyperplan séparateur. La dimension VC est un terme de capacité particulièrement bien adapté à ce genre de classificateur. En effet, nous verrons dans la suite qu'il est possible d'agir directement sur cette dimension en jouant sur la marge entourant l'hyperplan. Cette

[4]- en substituant directement ce dernier au coefficient d'éclatement

observation constitue une des motivations principales de la construction de classificateurs dits à marge maximale.

Définition 1.4 (dimension VC) La dimension VC d'un ensemble de fonctions , notée h est le nombre maximum de points pouvant être séparés de toutes les manières possibles par les fonctions de F .

Cela veut dire qu'il doit exister une configuration de h (= VC(F)) points, telle que les fonctions f ∈ F peuvent leur assigner les 2h combinaisons de labels possibles. A nouveau, cela n'est pas garanti pour n'importe quel ensemble de h points.

Pour concrétiser ces propos, considérons trois documents représentés dans $\mathbb{R}^2$. Supposons que la famille de fonctions f corresponde aux droites de $\mathbb{R}^2$

$$y = \gamma_1 x + \gamma_0$$

La dimension VC de F est 3 car on peut trouver une configuration de trois points séparables de toutes les façons possibles, par contre on ne peut trouver aucune configuration de 4 points (ou plus) rendant une telle discrimination possible.

Par extension, nous pouvons énoncer la proposition suivante:

Proposition 1 La dimension VC de l'ensemble des hyperplans de $\mathbb{R}^n$ est $n + 1$.

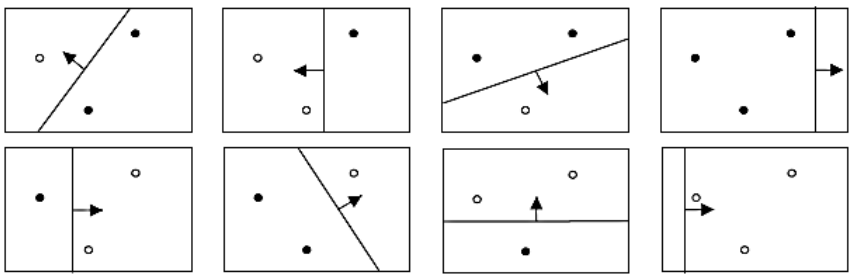


Figure 1.4- Exemple de configuration de 3 points séparables de toutes les manières possibles par les droites de $\mathbb{R}^2$. Les ronds vides et pleins représentent respectivement les points assignés positivement et négativement. La flèche représente le coté de la droite où les points seront classés positivement.

Sans rentrer dans les détails, pour construire une borne du type (1.22), nous avons besoin de formuler un terme qui prenne en compte le nombre d'exemples:

$$h(\ln\frac{n}{h}+1) \quad (1.23)$$

La partie gauche de la figure ci-dessous illustre le comportement de (1.22) en fonction du rapport n/h. Il apparaît clairement que lorsque le rapport est inférieur à 1, on se trouve dans une zone de croissance linéaire, alors que quet $n > h$, la fonction décroît de manière logarithmique. On préfère dès lors se placer dans la seconde partie, laquelle limite la contribution du terme de capacité dans l'expression de la borne sur le risque.

En substituant ce terme au terme relatif au coefficient d'éclatement dans (1.22) et en négligeant le facteur "8"[5], on obtient avec une probabilité $1 - \delta$:

$$R[f] \leq R_{emp}[f] + \sqrt{\frac{1}{n}(h(\ln\frac{2n}{h}+1) + \ln\frac{4}{\delta})} \quad (1.24)$$

Ce terme est appelé terme de confianceVC. La partie droite de la figure ci-dessous nous montre l'évolution de ce terme en fonction du rapport h/n. Ce tracé est effectué pour un ensemble de 30 000 documents avec δ fixé à 0.05. On observe que pour $n > h$, la fonction (terme de confiance) croît de manière monotone avec h. Par conséquent, une manière simple de réduire la contribution de ce terme dans la borne va être de minimiser h.

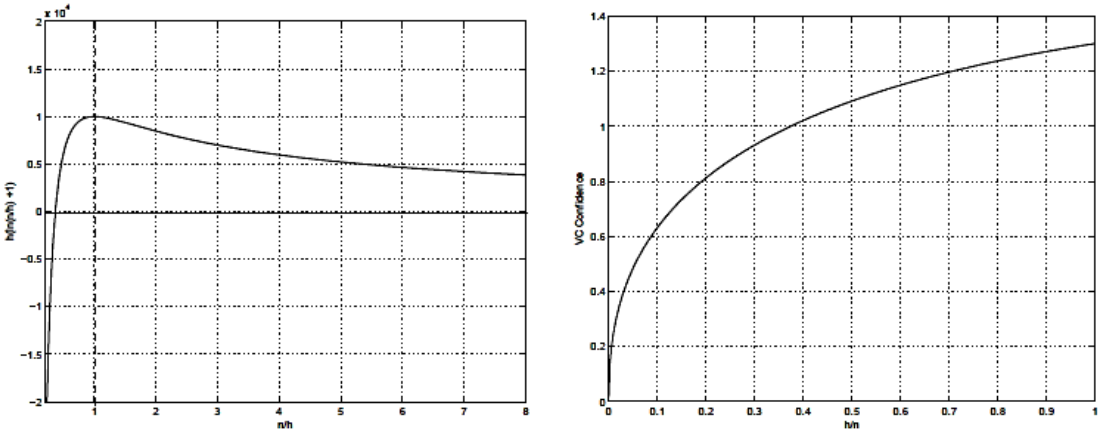


Figure1.5: Evolution du terme de Confiance VC.

[5]- chose que l'on peut faire en vertu de la limite tendant vers l'infini pour n

1.3.4 Minimisation du risque structurel (MRS)

Regardons à présent comment utiliser une borne du type (1.23) pour entraîner une machine d'apprentissage. Premièrement remarquons que dans ce type de borne, le risque empirique concerne une seule fonction alors que le terme de confiance est relatif à un ensemble de fonctions. L'idée de base va être de fixer une série de sous-ensembles de données, possédant chacun un terme de capacité propre, et ensuite d'effectuer un entraînement par MRE sur chacun d'entre eux. Concrètement, nous allons diviser l'ensemble F de la machine d'apprentissage en une structure qui consiste en l'imbrication successive de sous-ensembles dont la dimension VC va en augmentant (cf. figure 1.5). Une fois les entraînements sur chaque sous-ensemble terminés, nous allons déterminer le modèle optimal, en regardant quelle est la configuration qui minimise (1.23). On parle de sélection de modèle.

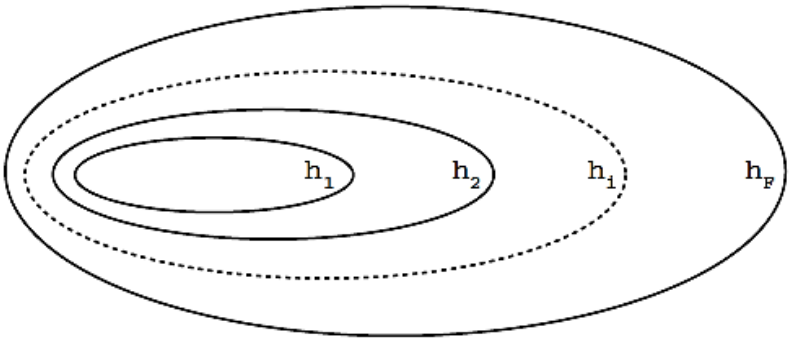


Figure 1.6 - Minimisation du risque structurel. Les sous-ensembles sont ordonnés selon leur dimension **VC** : $\mathbf{h_1} < \mathbf{h_2} < \ldots < \mathbf{h_i} < \ldots < \mathbf{h_F}$

1.3.5 Régularisation du risque

La régularisation permet d'adresser le dilemme biais-variance en formulant une fonction objective [6] qui fait intervenir l'erreur empirique et un terme pénalisant l'utilisation de fonctions appartenant à un ensemble de (trop) grande e capacité.

$$R_{reg}[f] = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}|f(x_i) - y_i| + \lambda P(f)$$

6-En termes d'optimisation mathématique, il s'agit de la fonction à optimiser.

Cette fonction objective s'appelle le risque régularisé. La constante λ permet de régler l'importance que l'on porte à un des deux pôles du dilemme. Cette constante est un paramètre d'entraînement qui peut être choisi par MRS. Comme nous le verrons, la méthode des SVMs (cf. chapitre 2) est particulièrement élégante quant à la formulation de son risque régularisé.

1.4 La classification en pratique

La définition de machine d'apprentissage ne nous indique nullement comment obtenir un classificateur adapté à la tâche considérée. Regardons à présent les différentes étapes que l'approche Machine Learning préconise pour atteindre un tel objectif.

1.4.1 Ensemble de données

En premier lieu, nous devons disposer d'un corpus Ω de documents qui ont déjà été classés selon les catégories qui nous intéressent. La catégorie peut être fournie par l'auteur du document mais dans la plupart des cas les documents sont classés à postériori par une personne appelée l'annotateur[7]. On considère de plus que les documents de Ω respectent les hypothèses prises sur le modèle génératif.

En pratique, on sépare le corpus en deux ensembles disjoints:

- le corpus d'apprentissage (training set (Tr)) : C'est à partir de ces données que le classificateur va être construit. On extrait quelquefois au préalable des documents du corpus d'apprentissage afin de constituer un ensemble permettant d'affiner les réglages: corpus de validation (validation set (Va)).

- le corpus de test (test set (Te)) : Ces documents vont être utilisés pour évaluer la performance du classificateur face à des données non-encore rencontrées jusqu'alors. La différence fondamentale qu'il y a entre le corpus de test et le corpus de validation, outre la taille restreinte de ce dernier, est le fait que les documents du test set ne peuvent en aucun cas être utilisés dans le processus inductif d'apprentissage.

[7] -Certains sites de presse comme l'agence Reuters mettent à disposition d'important corpus annotés

1.4.2 Entraînement ou apprentissage

La phase d'entraînement consiste à sélectionner une fonction $f \in F_\gamma$ c.à.d. à trouver une évaluation des paramètres γ_i. La sélection de ces paramètres est effectuée par un algorithme d'apprentissage qui reçoit en entrée le training set ainsi qu'un ensemble de paramètres d'apprentissage: $P_1, \ldots, P_k$. En ce sens, ce sont les données (du training set) qui induisent l'apprentissage. L'ensemble des paramètres, γ_i, résultant de l'apprentissage est appelé modèle. Une machine d'apprentissage munie d'un modèle est appelée machine entraînée. La figure 1.6 schématise un tel entraînement.

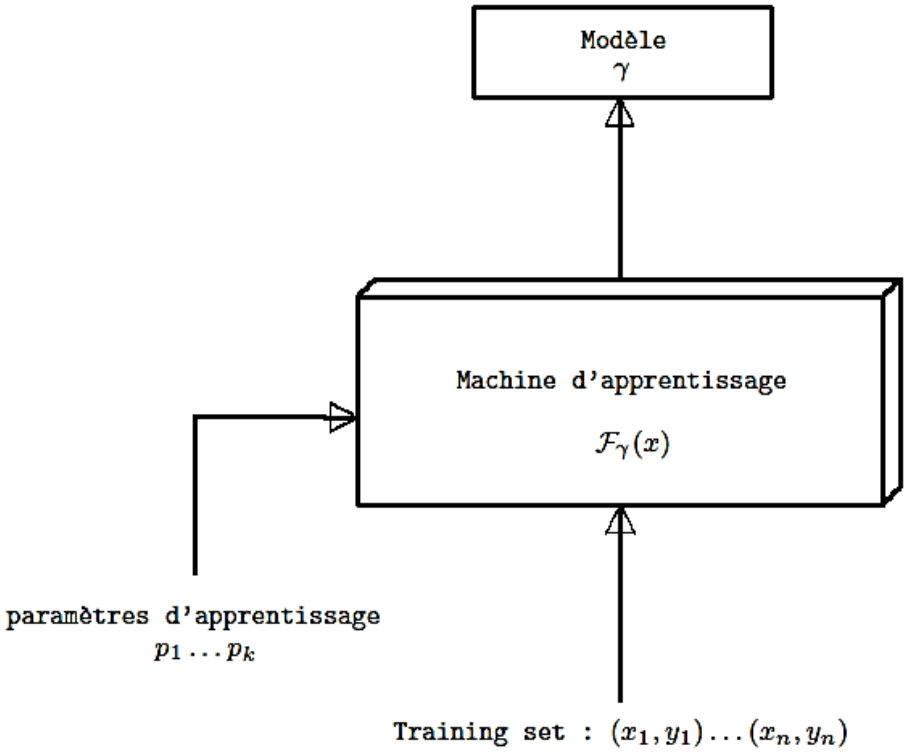


Figure 1.7 - Entraînement d'une machine d'apprentissage

1.4.3 Evaluation du modèle (test)

Une fois le modèle obtenu, il est intéressant d'évaluer ses performances sur un ensemble indépendant de données : le test set. En effet, on ne peut se fier aux résultats obtenus sur le training set car la machine d'apprentissage a perdu son indépendance face à ces données. Cette phase permet de se rendre compte du pouvoir de généralisation du

classificateur, ç.a.d. sa capacité à obtenir de bons résultats sur n'importe quel ensemble de données provenant de la même distribution.

Dans la pratique, certaines personnes observent comment évolue l'évaluation du modèle lorsqu'elles ajustent les paramètres d'apprentissage. Une telle pratique n'a bien entendu plus aucune rigueur scientifique puisque la machine d'apprentissage n'est plus indépendante du test set. Pour pouvoir effectuer une telle opération, il faut utiliser un autre ensemble indépendant: le validation set. On parle de phase de validation.

1.4.4 Exploitation

Lorsque l'on dispose d'un modèle efficace pour une tâche considérée, on peut utiliser la machine d'apprentissage pour faire des prédictions sur de nouveaux documents. Dans notre cas, un classificateur correspond donc à une machine entraînée. Les sociétés qui vendent ce type de classificateurs entraînent souvent la machine d'apprentissage sur un corpus relatif aux catégories spécifiées par le client. De cette manière, le client reçoit uniquement une machine entrainée sans avoir à se soucier de questions d'entraînement et de paramétrage. Certaines applications grand public telles que la reconnaissance vocale (Babel Speech) ou l'OCR (Read Iris, Omnipage) proposent également des classificateurs entraînés. Il y a souvent une phase d'adaptation (à la voix ou à l'écriture) dont le but est de sélectionner le modèle correspondant le mieux à l'utilisateur.

1.5 Quelques Méthodes de Classification

Les caractéristiques extraites de la forme permettent de définir un vecteur d'observation. Ces observations sont ensuite comparées à des références représentant les classes pour permettre d'associer une étiquette à chacune d'elles. Les références sont issues d'un apprentissage assuré grâce à un corpus de données pré-étiquetées (phase manuelle).

1.5.1 Approches statistiques

Chaque classe C_i est supposée être caractérisée par la loi à priori $P(C_i)$ et par la distribution des observations conditionnellement à la classe $P(y / C_i)$, où y est un vecteur d'observations ou une suite de vecteurs d'observations.

La décision d'appartenance de y à une classe est prise en recherchant la classe de probabilité à posteriori maximum :

$$P(C_i / y) = \max P(C_j / y) = \max P(y / C_j) P(C_j) / P(y) \qquad (1.25)$$

En supposant toutes les classes équiprobables, le critère de décision devient alors le critère du maximum de vraisemblance.

La classe identifiée C_i vérifie donc :

$$P(y / C_i) = \max P(y / C_j) \qquad (1.26)$$

Cette approche nécessite une phase d'apprentissage qui consiste à obtenir $P(C_i)$ et $P(y / C_i)$ à partir d'un ensemble d'échantillons.

Il existe différentes méthodes de classification que l'on peut classer en deux groupes :

- les méthodes paramétriques
- les méthodes non paramétriques

Ce classement se fait selon la connaissance à priori des lois de distribution des observations et des classes.

1.5.2 Méthodes paramétriques

Si on fait une hypothèse sur la forme de la loi de densité de probabilité $p(y / C_i)$, par exemple que l'on possède des informations initiales sur la « forme » de la classe, alors la loi générale est connue (par exemple une loi normale, une loi de Poisson, une loi gamma, …) et sa forme particulière est modulée par un nombre fini de paramètres.

L'apprentissage à l'aide des échantillons va simplement consister à trouver ces paramètres.

1.5.2.1 Le mélange de lois gaussiennes

La distribution probabiliste de chaque classe discriminée peut être modélisée par une gaussienne multidimensionnelle ou pour plus de précisions par un mélange de lois gaussiennes multidimensionnelles (MMG pour modèle de mélange de lois gaussiennes) . L'espace est considéré de dimension multiple.

Si y est une séquence de vecteurs (y_t), supposés indépendants alors :

$$P(y/C_i) = \prod_t P(y_t/C_i) \quad \text{avec} \quad P(y_t/C_i) = \sum_{l=1}^{p} \upsilon_l^i N(y_t, m_l^i, \Sigma_l^i) \tag{1.27}$$

où $N(y_t, m_l^i, \Sigma_l^i)$ représente une loi multi gaussienne

m_l^i représente le vecteur moyenne

Σ_l^i représente la matrice de covariance

υ_l^i représente les probabilités à priori

M désigne le nombre de classes du mélange

p désigne le nombre de loi multi gaussienne

Rappel concernant les densités correspondantes à des lois normales multidimensionnelles :

$$N(x, \mu_i, \Sigma_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu_i)^t \Sigma_i^{-1}(x-\mu_i)\right] \tag{1.28}$$

A titre d'illustration, la figure 1.7 présente un exemple en deux dimensions dans lequel on a représenté mille points issus d'un modèle de mélange comportant trois composantes.

On distingue les trois formes ellipsoïdales (qui n'ont d'ailleurs pas la même orientation, ce qui traduit le fait que les matrices de covariances sont différentes pour chaque composante du mélange). Il est clair que selon la manière dont les vecteurs moyens des composantes diffèrent, l'individualisation des composantes sera plus ou moins marquée.

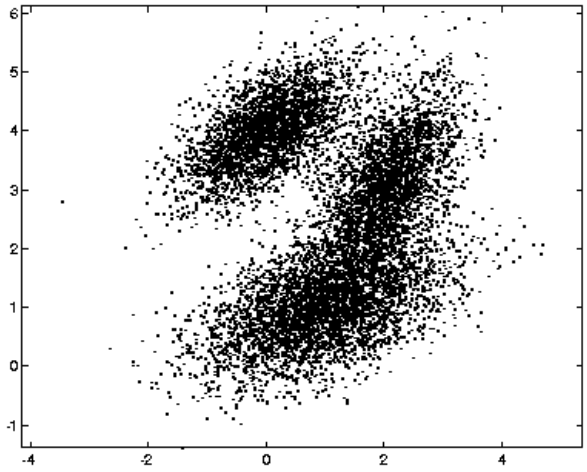


Figure 1.8: Exemple d'un mélange de trois densités gaussiennes bidimensionnelles.

En apprentissage, plusieurs approches sont possibles et entre autres :

- la méthode du maximum de vraisemblance

Les paramètres de la loi recherchée sont considérés comme « fixes » et le choix optimal est celui qui maximise la probabilité d'obtenir les échantillons fournis pour définir la classe.

- la méthode d'estimation Bayésienne

Les paramètres cherchés sont considérés comme des variables aléatoires ayant une distribution à priori connue.

1.5.3 Méthodes non paramétriques

Dans les méthodes d'apprentissage paramétriques, les échantillons permettent d'estimer les paramètres de la loi de densité de probabilité de chaque classe. Mais, toutes les lois de distributions classiques sont unimodales, elles ne peuvent donc prétendre représenter toutes les possibilités pour les classes. De plus, faire des hypothèses erronées sur le modèle à priori d'une classe peut amener à des résultats catastrophiques d'où la nécessité de l'apprentissage non paramétrique.

Les méthodes non paramétriques vont prendre en compte les échantillons et surtout leur répartition spatiale dans l'espace des paramètres pour obtenir des lois de densité de probabilités plus proche de la réalité de la classe.

Les décisions sont basées sur des statistiques locales (c'est à dire une estimation locale de la fonction de densité) comme la méthode des k plus proches voisins ou des histogrammes.

1.5.3.1 *Méthode de décision du plus proche voisin* :

Si l'on est capable de calculer la distance entre une forme x quelconque et un échantillon, alors la règle de décision du plus proche voisin peut être employée.

Le principe de décision consiste tout simplement à calculer la distance de la forme inconnue x à tous les échantillons fournis. La forme est alors classée dans la classe de l'échantillon le plus proche.

Par exemple dans la figure 1.8 ci-dessous x est classé dans la classe C2.

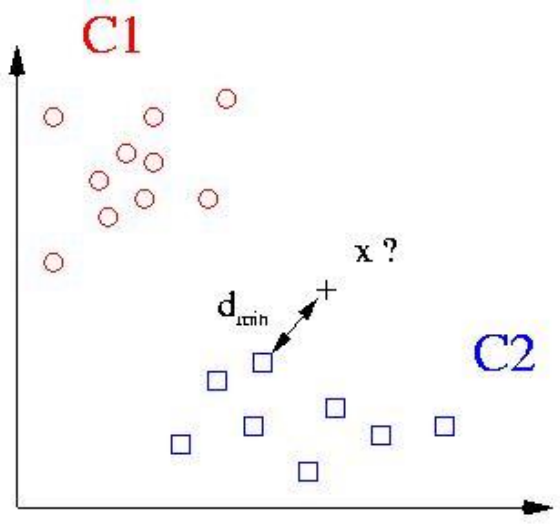


Figure 1.9 : Décision selon le plus proche voisin.

1.5.3.2 *Méthode de décision des k plus proches voisins* :

Il s'agit d'une extension de la méthode précédente (plus proche voisin) : pour une forme inconnue x à classer, on va calculer la distance de x à tous les échantillons, puis on sélectionne les k plus proches échantillons et on affecte x à la classe majoritairement représentée parmi ces k échantillons.

Dans l'exemple suivant (cf. figure 1.9), pour k=5 voisins, on classe x dans la classe C_1 alors que par la méthode du plus proche voisin il aurait été classé en C_2.

Il y a plusieurs choix possibles pour les distances :

- distance euclidienne
- distance « city block » (somme des valeurs absolues)
- distance de Tchebycheff (sup)

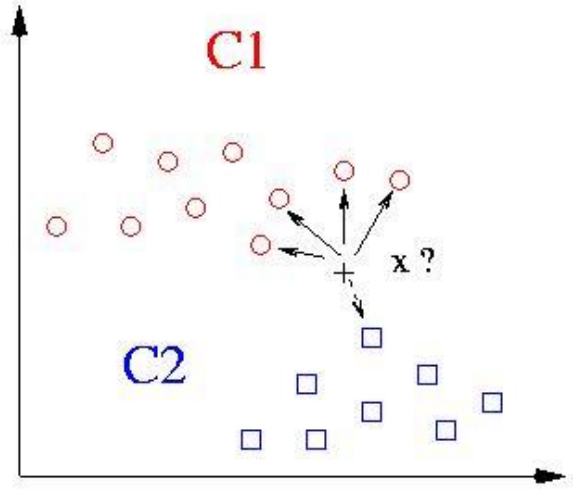
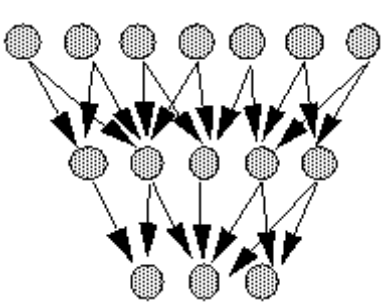


Figure1.10: Décision selon les k plus proches voisins.

1.5.4 Les Réseaux de Neurones

On distingue parmi les différents modèles neuronaux existants deux grandes classes d'algorithmes : celles qui utilisent l'apprentissage supervisé et celles qui déterminent leurs paramètres par apprentissage non supervisé. Dans la première classe, le modèle le plus connu et le plus utilisé est le Perceptron Multicouches qui permet de traiter les problèmes de régression non linéaire. Dans la seconde classe, on trouve le modèle des cartes

topologiques dont la première version est due à Kohonen. Utiliser les cartes topologiques permet d'aborder tous les problèmes classiquement connus en statistique sous le nom de classification automatique non supervisée. Un réseau de neurones est un ensemble d'éléments simples, reliés les uns avec les autres, qui se transmettent de l'information par l'intermédiaire de ces liens ou connections (cf. figure 1.10).



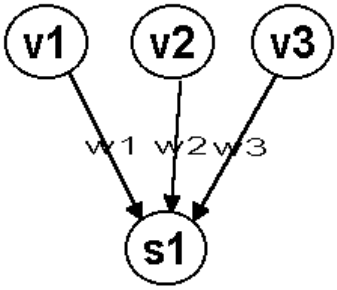


Figure 1.51 : Exemple de réseau de neurones. **Figure 1.6 :** Fonctionnement d'un neurone.

Chaque neurone additionne les valeurs de ses entrées (flèches entrant dans le neurone) et la renvoie aux suivantes (flèches sortantes) **(** cf. figure 1.11**).** Dans le cas de l'indexation sonore, l'entrée est formée de la suite d'observations et chaque sortie correspond à une classe **[Freeman et Skapura, 1991].**

Pour n entrées, la formule générale est la suivante :

$$S_1 = Seuil(\sum_{i=1}^{n} W_i V_i) \qquad (1.29)$$

La fonction Seuil assure que la valeur de s1 ne dépassera pas certaines limites "raisonnables" (en général l'intervalle 0-1). Elle interdit les évolutions "catastrophiques" (effets de boucle où les valeurs deviennent de plus en plus grandes).

Il existe plusieurs fonctions de seuil : les fonctions les plus "naïves" sont des seuils plus ou moins biseautés (cf. figure 1.12 Hard Limiter et Threshold Logic).

La fonction « Hard Limiter » est une fonction seuil avec I (entrée) :

- si I > seuil → neurone activé

- si I ≤ seuil → neurone inhibé

La fonction « Threshold Logic » est une fonction linéaire :

- si I > 1 → neurone activé
- si 0 ≤ I ≤ 1 → I
- si I < 0 → neurone inhibé

Cependant, les neurones à "sigmoïde" (cf. figure 1.12) sont beaucoup plus performants. Les fonctions sigmoïdes sont continues et en général bornées entre –1 et 1.

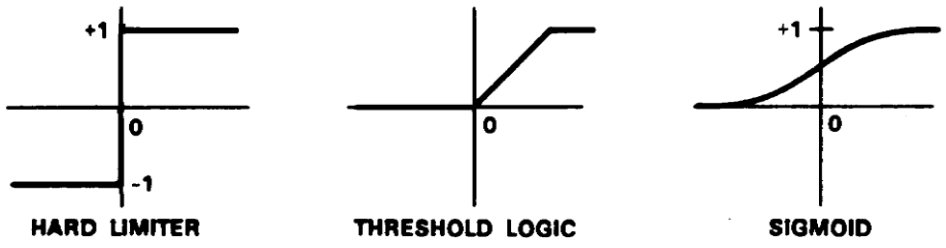


Figure 1.7: Quelques fonctions de seuils.

Les perceptrons sont des réseaux sans contre réaction : les sorties des neurones de la couche i forment les entrées de la couche i+1. La figure 1.13 montre la structure d'un perceptron à trois couches dont une couche cachée, appliqué à un problème de reconnaissance globale de mots. Ce modèle est issu des travaux de F. Rosenblatt sur le perceptron monocouche, plusieurs travaux concernant la reconnaissance de l'écriture arabe sont effectués en utilisant cette dernière architecture (cf. **[Farah et al 2005, Souici et Sellami, 2002]).**

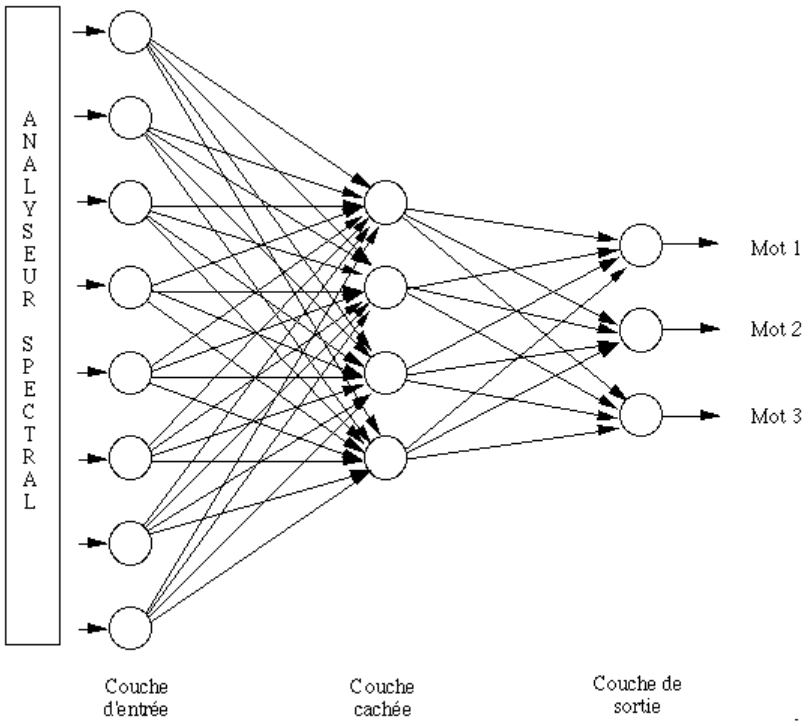


Figure 1.8 : Structure d'un perceptron à trois couches.

1.5.5 Classification par Modèle de Markov Caché

Un Modèle de Markov Caché (MMC) ou HMM (Hidden Markov Model) est semblable à une machine à états finis dans laquelle les transitions et les résultats sont stochastiques. Les MMC sont généralement utilisés par les systèmes de reconnaissance de l'écriture **[Gasmi et al 2005, Hassini et al, 2004]** pour faciliter l'identification des mots représentés par les ondes sonores captées. Dans ce cas, un MMC décrit la réalisation d'une concaténation de processus élémentaires qui représentent la séquence de paramètres acoustiques extraits d'un énoncé humain. Un MMC est donc une représentation statistique d'un événement.

La classification consiste à définir pour chaque classe un Modèle de Markov Caché, un automate d'états finis probabilisé, où chaque état représente un son et émet de manière probabiliste un vecteur d'observations. Les transitions représentent les différentes possibilités d'enchaîner les sons. La fonction de décision est basée sur le critère du maximum de vraisemblance.

De manière synthétique, un MMC est donc un automate pondéré stochastique particulier capable, après avoir été entraîné, d'estimer la probabilité qu'une séquence d'observations donnée ait pu être générée par son modèle. Dans l'absolu, à chaque phrase doit correspondre un modèle. Mais ceci est, évidemment, irréalisable. Des sous unités de la phrase, telles que le phonème, la syllabe ou le mot, sont associées à un modèle de Markov caché de manière à réduire le nombre de paramètres à entraîner. Chacun de ces modèles est constitué d'un nombre fini d'états prédéterminés

Paramétrage d'un MMC

- Partie markovienne
    - Spécifier la distribution initiale Π = (Π1,…, ΠN)'.
    - Spécifier la matrice de transition A = [a_{ij}]. A est une matrice stochastique c'est à dire que $\sum_{j=1}^{N} a_{ij} = 1$ avec 0 ≤ i, j ≤ N.
- Partie observation
    - La matrice d'observation notée B = [bi,y] caractérise la loi d'observation sachant que y ∈ ΩT (ensemble des observations).

Pour l'exemple de la figure 1.13, les états 1, 2 et 3 sont des états émetteurs c'est à dire qu'on a une observation associée chaque fois que l'on arrive dans ces états.

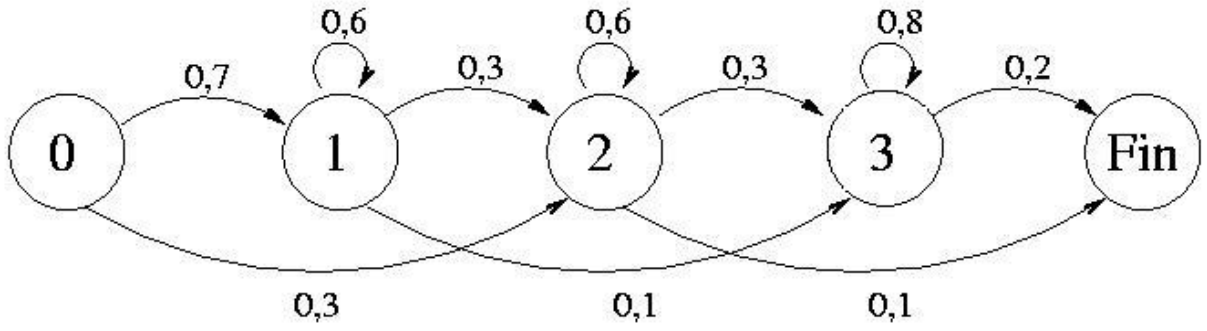


Figure 1.9: Exemple de Modèle de Markov Caché.

Conclusion

Quelque soit la méthode de décision employée, un ensemble d'apprentissage est nécessaire, afin d'apprendre les seuils, les caractéristiques des classes ou les paramètres des modèles statistiques (histogrammes, moyennes, variances, paramètres des MMC ou des Réseaux de neurones).dans le prochain chapitre nous discutons la méthode de décision par SVMs. Les performances des classificateurs sont, comme dans de nombreux problèmes de reconnaissance des formes, extrêmement dépendants de la qualité de cet apprentissage.

Chapitre II :

Les Machines à Vecteurs de Support (SVM).

1.1 Introduction

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

En 1995, Cortes et al. proposent une version régularisée des SVMs qui tolère les erreurs d'apprentissage tout en les pénalisant **[Cortes et Vapnik, 1995].**

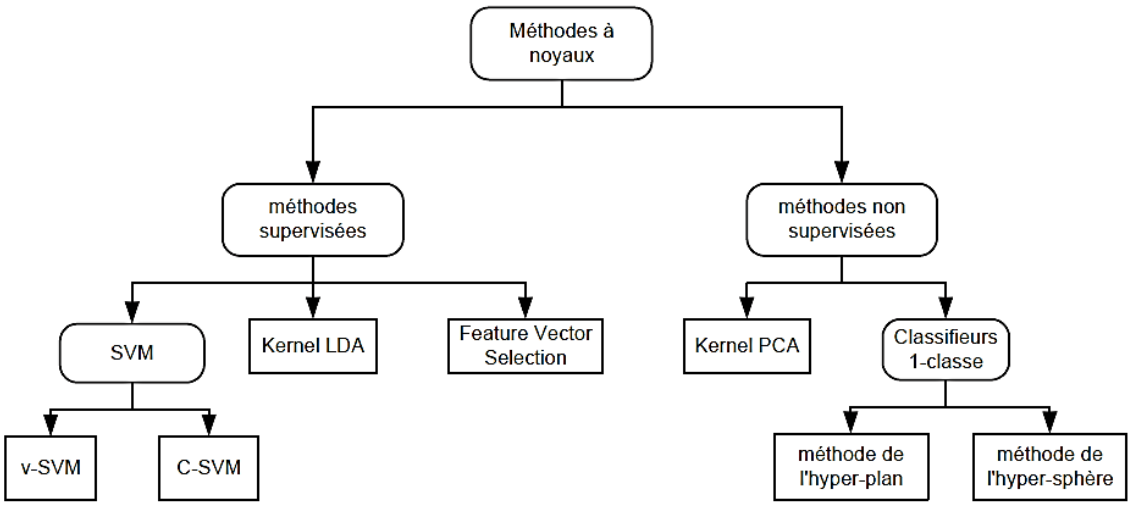


Figure 2. 1. : Arbre de classification des méthodes d'apprentissage à base de noyaux

Depuis, les SVMs (*le pluriel est utilisé pour désigner les différentes variantes du SVM*) n'ont cessé de susciter l'intérêt de plusieurs communautés de chercheurs de différents domaines d'expertise. Par exemple, Cortes et al. dans **[Cortes et Vapnik,1995]**, Scholkopf et al. [**Scholkopf et al, 1999**] et Burges et al. **[Burges et al, 1997.]** ont appliqué les SVMs à la reconnaissance de chiffres manuscrits isolés, Blanz et al. **[Blanz et al,1996.]** ont expérimenté les SVMs sur des objets 2D de vues différentes. Schmidt et

al. **[Schmidt, 1996.]** ont exploré la tâche de reconnaissance d'orateur. Osuna et al. ont traité la reconnaissance d'images de visages **[Osuna et al, 1997]**.

Dans la plupart des cas, la performance des SVMs égale ou dépasse celle de modèles classiques déjà établis. L'estimation de densité de probabilité **[Weston, et al, 1997]** et la décomposition ANOVA **[Stitson, et al, 1997]** ont été aussi explorées. D'autres auteurs ont aussi étudié l'apport de la connaissance à priori pour ce type de classificateurs. Ainsi, les SVMs virtuel de Burges et al. **[Burges et al, 1997.]** améliorent la généralisation en appliquant un bruit spécifique à l'ensemble de vecteurs de support **[Chris et al, 1997].**

Smola et al. ainsi que Wahba ont mis en évidence la ressemblance entre le SVM et la théorie de régularisation **[Smola, 1998], [Girosi, 1997], [Wahba, 1998]**. Ils ont démontré qu'associer un noyau particulier à un SVM revient à considérer une pénalisation différente de l'erreur d'apprentissage en maximisant la marge. Ce qui nous permet de dire que la maximisation de la marge dans l'espace augmenté est une forme de régularisation de l'apprentissage. Dès lors, les SVMs permettent de répondre à deux problèmes centraux de la théorie de l'apprentissage statistique que sont:

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

1.2 Risque structurel

Dans cette section, nous rappelons quelques éléments essentiels de la théorie de l'apprentissage statistique. Nous introduisons par la même occasion le principe du risque structurel que minimise le SVM **[Vapnik, 1995].**

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

$$(x_1, y_1), \ldots\ldots (x_1, y_1) \in \mathbb{R}^N \mathrm{x}\{\pm 1\} \qquad (2.1)$$

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

Et soit l'ensemble de fonctions f_α défini:

$$\{f_\alpha : \alpha \in \Lambda\} \in \mathbb{R}^N \rightarrow \{\pm 1\} \qquad (2.2)$$

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

$$R(\alpha) = \int dP\,(x, y). \qquad (2.3)$$

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

$$R_{emp}(\alpha)\frac{1}{l}\sum_{i=1}^{l}\frac{1}{2}|\,f_a(x_i) - \,y_i| \qquad (2.4)$$

Si l'ensemble d'apprentissage est fini, la minimisation du risque empirique donné par l'équation 2.4 ne garantit pas un minimum pour le risque réel.

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

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

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

$$R(\alpha) \leq R_{emp}(\alpha) + \emptyset(\frac{h}{l}, \frac{\log(\eta)}{l}) \qquad (2.5)$$

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

$$\phi\left(\frac{h}{l}, \frac{\log(\eta)}{l}\right) = \sqrt{\frac{h\left(log\frac{2l}{h}+1\right)-\log(\frac{\eta}{4})}{l}} \qquad (2.6)$$

Le paramètre h est appelé VC ou dimension de **V**apnik-**C**hervonenkis (Voir ChapitreI, Section1.3.3). Il décrit la capacité de l'ensemble de fonctions solutions. Pour un problème de classification binaire, h est le nombre maximum de points séparables selon 2^h

configurations par l'ensemble de fonctions solutions. La minimisation de cette borne consiste à opérer une régularisation de l'erreur empirique $R_{emp}(\ a)$ via le terme de pénalité $\phi(\frac{h}{l}, \frac{log\eta}{l})$. Donc un minimum est garanti en minimisant à la fois l'erreur empirique et le terme de régularisation.

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

La limite supérieure du risque réel (erreur de généralisation) définie dans l'équation 2.5 constitue un principe essentiel de la théorie des Machines à Vecteurs de Support qui nécessite encore quelques remarques pertinentes.

Selon l'équation 2.5, étant donné un ensemble d'apprentissage de taille l, il est possible de borner $R(a)$ en minimisant la somme des quantités: $R_{emp}(a)\ et\ h(\{f_a : \ a\ \in Á\})$, tel que $Á$ représente un sous-ensemble de Λ. Par ailleurs, l'erreur empirique dépend de la solution particulière trouvée par la machine d'apprentissage, à savoir f_α et peut être réduite en choisissant des valeurs appropriées pour les paramètres a_i composant a.

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

$$S_1 \subset S_2 \subset \cdots \ldots \subset S_n \subset \cdots, \qquad (2.7)$$

et dont les capacités respectives vérifient l'inégalité:

$$h_1 \leq h_2 \leq h_n \leq \ldots \qquad (2.8)$$

Pour un ensemble observations $(x_1, y_1) \ldots, (x_1, y_1)$, la minimisation du risque structurel vise à choisir f_a parmi l'ensemble de fonctions $\{f_a : \ a\ \in \Lambda_n\}$ possibles, qui minimise le risque garanti (le risque garanti a été utilisé par Guyon et Boser pour désigner la limite supérieure de l'erreur de généralisation donnée dans l'équation 2.5).

Le processus de choisir le bon sous-ensemble de fonctions solutions revient à contrôler la complexité du c1assifieur en cherchant le meilleur compromis entre une faible erreur empirique et une complexité moindre. Notons enfin, que plusieurs travaux ont abouti à des résultats similaires quant à la nécessité d'un compromis entre l'erreur d'apprentissage et la complexité du modèle pour assurer une faible erreur de généralisation. Il en est ainsi

avec la théorie de la régularisation, le Minimum Description Length **[Rissanen, 1978], [Kolmogorov,1965]** et le dilemme Biais-Variance **[Geman et al,1992].**

2.3 Espace augmenté

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

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

$$\Phi: \mathbb{R}^2 \rightarrow F = \mathbb{R}^3 \qquad (2.9)$$

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

$$(x_1, x_2) \longrightarrow ((x_1^2, x_2^2, x_1, x_2)$$

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

$$N_F = \frac{(N+d-1)!}{d!(N-1)!}, \quad (2.10)$$

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

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

2.3.1 Noyau polynomial

Pour pouvoir représenter des produits scalaires de la forme $\Phi(x).\Phi(y)$. (voir équation 2.9), nous utilisons la notation:

$$k(x,y) = \Phi(x).\Phi(y), \qquad (2.11)$$

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

Ce calcul ne nécessite pas le calcul des images $\Phi(x).\Phi(y)$ explicitement. Cette propriété

a été utilisée par Boser, Guyon et Vapnik pour étendre l'algorithme de l'hyper-plan généralisé au SVM non-linéaire **[Boser et al, 1992.].** L'espace augmenté F est alors appelé espace de linéarisation (cf. Figure 2.2).

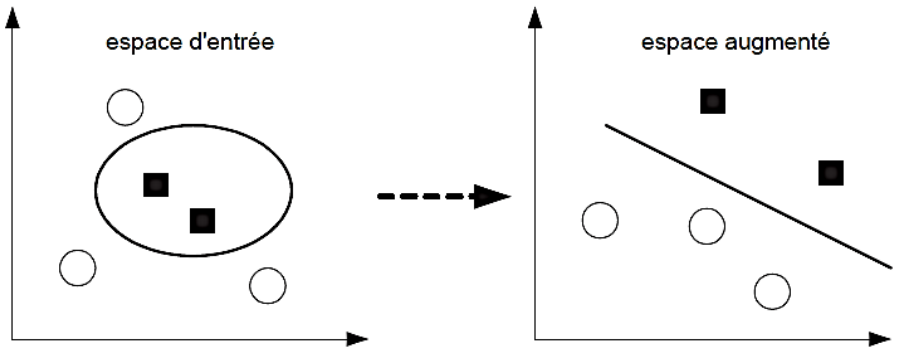


Figure 2. 2. : Transformation des données avec les noyaux de Mercer

En réalité, Aizerman et al. étaient les premiers à avoir élucider le concept de manipulation de produits scalaires via un noyau non-linéaire $k(x, y)$ **[Aizerman et al, 1964]**. Tout noyau décomposable selon la forme donnée dans l'équation 2.11 est un noyau Mercer **[Courant et Hilbert, 1953]**. On démontre que tout noyau continu, symétrique et semi-défini positif est un noyau de Mercer.

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

$$C_2(x): (x_1, x_2) \rightarrow (x_1^2, x_2^2, x_1, x_2, x_1, x_2),$$

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

$$(C_2(x)C_2(y)) = x_1^2, y_2^2 + x_1^2, y_2^2 + 2x_1x_2, y_1, y_2 = (\mathrm{x.y})^2$$

Cet exemple simple montre qu'un noyau quadratique produit un espace augmenté implicite dans $\mathbb{R}^4$. Plus généralement, il est démontré qu'un noyau polynomial d'ordre d projette les données dans l'espace des monômes d'ordre d **[Poggio, 1975].** Une autre fonction Φ possible pour ce même noyau aurait pu être:

$$\Phi(x) = x_1^2, x_2^2, \sqrt{2}\, x_1x_2) \qquad (2.12)$$

Le tableau 2.1 montre quelques noyaux de Mercer classiques utilisés pour le SVM.

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

Tableau 2.1. : Quelques noyaux classiques

2.4 Formulation

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

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

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

$$E(y, f(x) = \begin{cases} 0\ si\ y = f(x) \\ 1\ sinon \end{cases} \qquad (2.14)$$

2.4.1 Hyper-plans de séparation

Nous avons énoncé dans l'équation 2.7 que des structures imbriquées S_n d'espaces de fonctions mettent en oeuvre le principe de la minimisation du risque structurel si un compromis entre la complexité du modèle et une faible erreur d'apprentissage est trouvé. L'algorithme de construction du SVM considère des hyper-plans de séparation entre les classes à discriminer.

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

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

$$\{z \epsilon F : (w_0 . z) + b_0\}, \qquad (2.15)$$

où $(w, b) \in F$ x $\mathbb{R}$.

Cette formulation permet toujours de choisir des surfaces dont le couple (w, b) est un multiple de (w_0, b_0) de l'équation 2.15. Cependant, il est d'usage de considérer une formulation canonique telle que:

$$min_i = 1, \ldots, l | (w . z_i) + b) | = 1, \qquad (2.16)$$

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

$$D = \frac{2}{\|w\|} \qquad (2.17)$$

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

$$f_{w,b} = sgn((w . z) + b) \qquad (2.18)$$

vérifie:

$$h < \mathcal{R}^2 \mathrm{A}^2 + 1 \qquad (2.19)$$

Dans l'équation 2.19, R est le rayon de la plus petite hyper-sphère englobant les données d'apprentissage $z_1, \ldots . z_1, et\ A$ un scalaire tel que $A \geq 2 / D$. Si $\|w\| > A$, la borne supérieure de la dimension VC devient $N_F + 1$, où N_F est la dimension effective de

l'espace augmenté F.

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

2.4.2 Hyper-plans à marge optimale

Supposons que nous ayons un ensemble d'observations $(z_1, y_1), \ldots, (z_1, y_1)$, tel que $z_i \in F$

$et\ y_i \in \{\pm 1\}$.

Idéalement, nous cherchons à trouver la fonction de décision

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

telle que:

$$f_{w,b}(z_i) = y_i, i = 1, \ldots . l \qquad (2.20)$$

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

$$y_i\ ((z_i.w) + b) \geq 1, i = 1, \ldots, l. \qquad (2.21)$$

Le lagrangien primaire s'écrit alors

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i (y_i\big((z_i.w) + b\big) - 1) \qquad (2.22)$$

Les variables α_i correspondent aux facteurs de Lagrange des contraintes définies dans l'équation 2.22. Au minimum de la fonction objective $L(w, b, \alpha)$, ses gradients par rapport à $w\ et\ b$ deviennent:

$$\frac{\partial}{\partial b} L(w, b, \alpha) = 0 \qquad (2.24)$$

Résoudre les équations 2.23 et 2.24

$$\frac{\partial}{\partial w} L(w, b, \alpha) = 0 \qquad (2.25)$$

donne:

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

avec

$$\alpha_i \left[y_i \left((z_i . w) + b \right) - 1 \right] = 0, i = 1, \ldots . , l \quad (2.26)$$

En remplaçant w par son expression de l'équation 2.25 dans l'équation 2.22, et en prenant en compte la contrainte de l'équation 2.26, la fonction objective $L(w, b, \alpha)$ prend la forme d'un Lagrangien dual à maximiser fonction de α_i seulement. Il s'écrit:

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=j}^{l} \alpha_i \alpha_i y_i y_i \ (z_i . z_i) \qquad (2.27)$$

avec les contraintes :

$$\alpha \geq 0, i = 1, \ldots \ldots l \qquad (2.28)$$

Et

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \qquad (2.29)$$

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

$$f(x) = sgn(\sum_{i=1}^{l} \alpha_i y_i (z . z_i) + b) \qquad (2.30)$$

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

2.4.3 Hyper-plans à marge molle

Le problème d'optimisation quadratique énoncé dans l'équation 2.27 a une solution dans le cas de données séparables uniquement. Dans le cas contraire, les conditions de Tucker (KKT) ne sont jamais satisfaites.

Cortes et al. **[Cortes et Vapnik ,1995]** utilisent une technique qui consiste à accepter des erreurs d'apprentissage tout en les pénalisant. Les auteurs ont introduit un paramètre de

pénalisation C qui règle le degré de compromis désiré entre la séparabilité des classes et l'étanchéité du modèle aux erreurs d'apprentissage. Pour ce faire, redéfinissons la contrainte de l'équation 2.21 en considérant des variables :

$$a\xi \geq 0, i = 1, \ldots . . l,$$

associées aux données $z_i \in F, \forall_i = 1, \ldots, l$ de l'ensemble d'apprentissage.
La variable ξ_i est la distance séparant z_i de la frontière de la marge qui est définie$\xi[1 - f(z)y]_+, pour\ i = 1, \ldots . , l$ L'opérateur $[\ ]_+$ est défini :

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

La contrainte de l'équation 2.21 devient alors:

$$y_i\big((z_i.w) + b\big) \geq 1 - \ \xi_i, i = 1, \ldots, l \qquad (2.31)$$

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

$$L(w, \xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i \qquad (2.32)$$

où C représente le facteur de Lagrangien associé au terme de pénalisation rajouté.
Cortes et Vapnik démontrent **[Cortes et Vapnik, 1995]** que la fonction de décision dans ce cas est caractérisée
par:

$$w = \sum_{i=1}^{l}\alpha_i y_i z_i \qquad (2.33)$$

et que le Lagrangien dual à maximiser s'écrit:

$$W(\alpha) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j}^{l}\alpha_i\alpha_j y_i y_j (z_i.z_j) \qquad (2.34)$$

avec les contraintes :

$$0 \leq \alpha_i \leq C, i = 1,\ldots..l, \qquad (2.35)$$

et

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \qquad (2.36)$$

2.4.4 Le SVM non-linéaire

Boser at al**. [Boser et al, 1992],** ont su produire des frontières de décision non-linéaire avec le SVM (cf .Figure 2.4). L'idée est d'utiliser un noyau de Mercer qui permet de projeter les données dans un espace éventuellement plus grand dans lequel une séparation linéaire des classes est possible **[Courant et Hilbert, 1953].** Il est alors important que la formulation du produit scalaire dans l'équation 2.27 reste intacte après l'introduction du noyau. Boser et al. démontrent qu'un noyau de Mercer ayant la forme:

$$\emptyset(x).\emptyset(y) = k(x, x_i) \qquad (2.37)$$

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

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

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j}^{l} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \qquad (2.38)$$

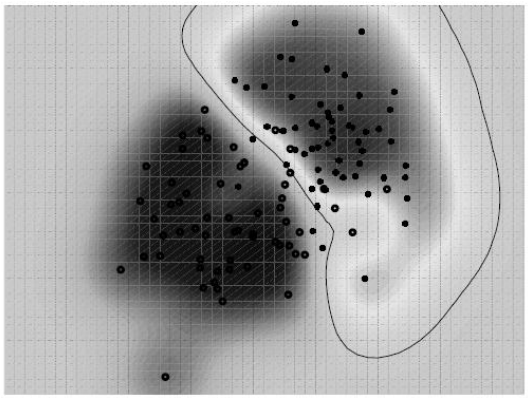


Figure 2. 3. : Frontière de décision non linéaire

La fonction de décision s'écrit encore :

$$F(x) = sgn(\sum_{i=1}^{l} y_i \alpha_i k(x, x_i) + b) \qquad (2.39)$$

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

2.4.4.1 *SVM L2*

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

$$L(w, \xi) = \frac{1}{C}\frac{\|w\|^2}{2} + \frac{1}{2}\sum_i \xi_i^2 \qquad (2.40)$$

qui équivaut à la maximisation de la fonction objective:

$$W(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \tilde{k}(x_i, x_j), \qquad (2.41)$$

lorsque:

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

$$\tilde{k}(x_i, x_j) = k(x_i, x_j) \quad if \quad i \neq j$$

avec les contraintes

$$\sum_i \alpha_i y_i = 0$$

et

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

2.4.5 Conditions de Karush-Kuhn-Tucker

La résolution de l'optimisation quadratique de l'équation 2.38 est basée sur les conditions de convergence dites de «Karush- Kuhn- Tucker» (KKT) qui établissent les conditions nécessaires (mais parfois suffisantes) de convergence de la fonction objective duale. Ces conditions sont relativement simples et s'écrivent:

$$\begin{aligned} \alpha_i = 0 &\Rightarrow y_i f(x_i) \geq 1 \quad et\ \xi_i = 0 \\ 0 < \alpha_i < C &\Rightarrow y_i f(x_i) = 1 \quad et\ \xi_i = 0 \\ \alpha_i = C &\Rightarrow y_i f(x_i) \leq 1 \quad et\ \xi_i \geq 0 \end{aligned} \qquad (2.42)$$

Les équations 2.42 reflètent une propriété importante du SVM stipulant qu'une grande proportion des exemples d'apprentissage sont situés en dehors de la marge et ne sont pas retenus par le modèle. Par conséquent, leurs multiplicateurs α_i sont nuls.

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

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

2.4.6 Calcul du biais b

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

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

est alors vérifiée.

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

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

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

2.4.7 Le v-SVM

Plusieurs modifications ont été proposées pour la formulation de base des SVMs. Parmi celles-ci, le v -SVM initialement prévu pour la régression mérite de s'y attarder **[Scholkopf et al, 2000].**

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

$$L(w, \xi, v, \rho) = \frac{1}{2}\|w\|^2 - v\rho + \frac{1}{l}\sum_{i=1}^{l} \xi_i \qquad (2.43)$$

avec les contraintes :

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

Les auteurs démontrent que $v\ (0 \leq v \leq 1)$ est une borne supérieure de l'erreur d'apprentissage et une borne inférieure du nombre de vecteurs de support. Cette propriété permet de contrôler à la fois l'erreur d'apprentissage et la complexité du c1assifieur à travers le nombre de vecteurs de support.

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

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

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

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

$$\sum_{i=1}^{l} \alpha_i \geq v$$

2.5 Discriminant de Fisher non-linéaire (KFD)

L'idée du discriminant de Fisher non-linéaire (de l'anglais «Kernel Fisher Discriminant») est de résoudre le discriminant de Fisher classique dans l'espace augmenté *F* induit avec un noyau de Mercer quelconque **[Fisher, 1936]**. La discrimination devient non-linéaire dans l'espace d'origine **[Mika et al , 1999], [Roth et Steinhage., 1999**

. Le discriminant de Fisher désigné par LDA (de l'anglais «Linear Discriminant Analysis»)

vise à trouver l'hyper-plan qui maximise la variance inter-classe et minimise la variance intra-classe des données (cf. Figure 2.4). Ceci est traduit par le critère suivant:

$$J(w) = \frac{w^T S_B w}{w^T S_W w} \qquad (2.44)$$

où

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

représente la variance inter-classe et

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

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

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

$$w = \sum_{i=1}^{l} \alpha_i \emptyset(x_i) \qquad (2.45)$$

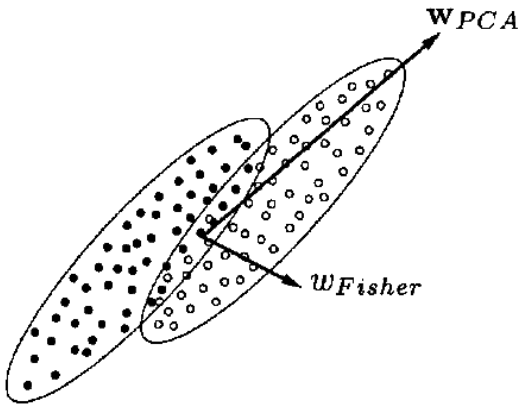


Figure2. 4 : Discriminant de Fisher: $\boldsymbol{W_{Fisher} \; et \; W_{PCA}}$ sont respectivement le vecteur directionnel du discriminant de Fisher et l'axe principal du nuage de points

l étant le nombre total d'exemples.

En substituant l'équation 2.45 dans l'équation 2.44, nous trouvons le critère d'optimisation

à maximiser suivant:

$$J(\alpha) = \frac{(\alpha^T \mu)^2}{\alpha^T N \alpha} = \frac{\alpha^T M \alpha}{\alpha^T N \alpha} \qquad (2.46)$$

où

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

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

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

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

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

Dans le but de solutionner l'équation 2.46, on résout le système propre donné par

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

dont la solution est $\alpha = N^{-1}(\mu_2 - \mu_1)$. Toutefois, pour un grande ensemble de données, la solution est prohibitive en mémoire et en temps de calcul. Mika et al. **[Mika et al, 2001].**

[Mikaet al, 1999] ont pu transformer la fonction objective $J(\alpha)$ en un problème de programmation linéaire convexe rendant possible sa résolution.

2.6 Analyse en Composantes Principales non-linéaire

L'Analyse en Composantes Principales (ACP) est une technique descriptive qui calcule les axes d'inertie d'un nuage de points (l'axe W_{PCA} dans la figure 2.6 représente l'axe principal des points). Les axes d'inertie représentent des directions orthogonales suivant lesquelles la variance des données est maximale. Le terme «composante principale» est l'association d'une direction principale et de son amplitude. Les composantes principales est une solution au sens des moindres carrés qui revient à procéder à une rotation de l'ensemble des données suivie d'une translation. L'ACP peut être utilisée pour réduire la dimensionnalité des données en conservant uniquement un sous-ensemble des composantes principales. Toutefois, l'ACP est une méthode linéaire qui ne détecte pas les structures non-linéaires des données.

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

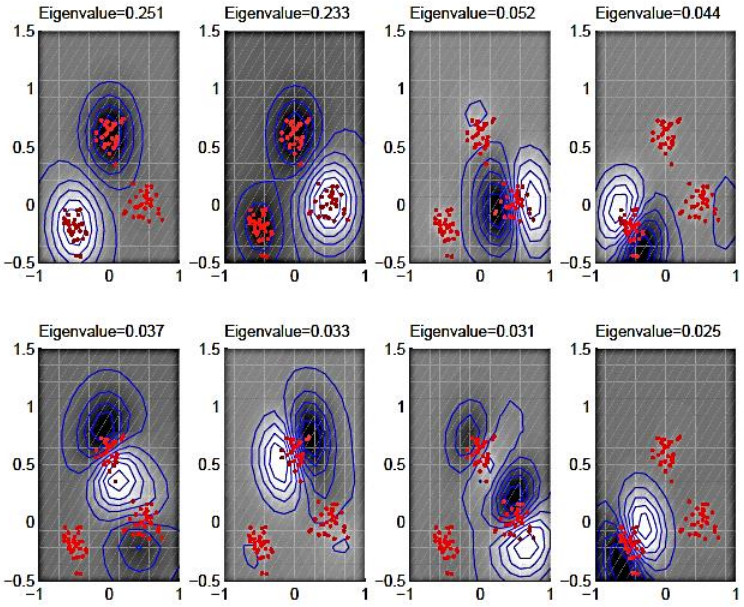


Figure 2. 5. : Découverte de structures par projection des données sur les huit premières composantes principales du KPCA

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

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

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

$$Trouver\ \lambda > 0, \boldsymbol{V} \neq 0\ avec$$

$$\lambda \boldsymbol{V} = C\boldsymbol{V} = \frac{1}{l}\sum_{j=1}^{l}(\phi(x_j).\boldsymbol{V})\phi(x_j) \qquad (2.47)$$

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

$$V = \sum_{i=1}^{l} \alpha_i \phi(x_i) \qquad (2.48)$$

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

système propre :

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

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

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

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

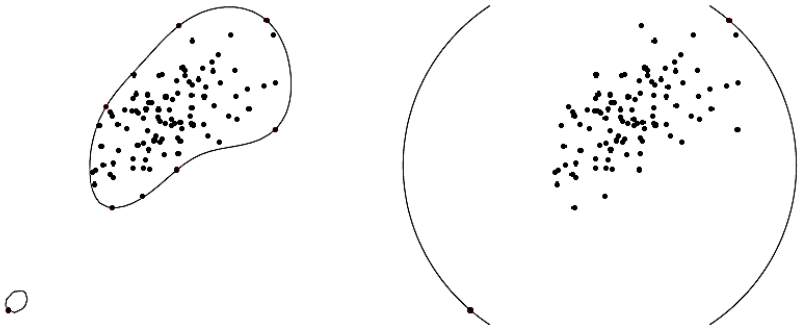
Il est utile de signaler que cette version de l'ACP donne lieu aux composantes principales au lieu de N dans l'ACP classique (N étant la dimension de l'espace de d'entrée). Celles-ci deviennent difficiles à calculer dès que nous disposons d'une grande quantité de données.

2.7 Classification mono-classe

Les méthodes d'estimation de densité sont des techniques d'apprentissage non-supervisées. Cette estimation peut s'avérer problématique pour plusieurs raisons. D'abord, il n'existe pas toujours une densité (quelques distributions n'ont pas de densité de probabilité). Aussi, estimer avec précision cette densité n'est pas garantie. Par ailleurs, dans plusieurs applications, il est suffisant de calculer le support de la distribution de données. Le SVM monoclasse a été proposé pour la première fois par Vapnik pour l'estimation du contour d'une distribution de points **[Vapnik, 1995].** Quelques années plus tard, deux autres algorithmes utilisant la théorie des noyaux ont été proposés.

2.7.1 Méthode de l'hyper-sphère

Cette méthode, proposée par [**Tax et Duin, 1999]**, isole les données dans une hyper-sphère de l'espace augmenté F **.** La frontière obtenue dans l'espace d'entrée est non-linéaire et dépend du noyau utilisé (cf. Figure 2.7.).



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

Figure 2. 6. : Surface de l'hyper-sphère englobant dans l'espace d'entrée pour deux types de noyaux

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

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

sous les contraintes :

$$\sum_{i=1}^{l} \beta_i = 1 \forall\, i \quad \beta_i \geq 0$$

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

$$\sum_{i=1}^{l} \beta_i \emptyset\,(x_i)$$

Les observations de l'ensemble des données pour lesquelles la variable β_i est différente de zéro sont les vecteurs de support de l'hyper-sphère. L'équation de l'hyper-sphère dépend de ce sous-ensemble de données. Ces points sont situés sur la surface de l'hyper-sphère et par conséquent sur la frontière de séparation qui isole les données (cf. Figure 2.7.).

Une version régularisée de l'algorithme permet d'isoler les données à l'intérieure de 1'hyper-sphère tout en permettant quelques exemples d'être en dehors de celle-ci. Cependant, un facteur γ est introduit pour pénaliser l'éloignement de ces points de la surface de la boule. Dans ce cas, il existe une contrainte supplémentaire $\beta_i < \gamma$ **[Tax et Duin, 1999].**

2.7.2 Méthode de l'hyper-plan

Cette méthode a été proposée par Scholkopf et al **[Scholkopf et al, 1999]**. et vise à isoler les images des points dans F par un hyper-plan au lieu d'une hyper-sphère. Moyennant un noyau non linéaire, cet hyper-plan donne lieu à des contours fermés dans l'espace d'entrée.

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

$$\frac{1}{2}\|w\|^2 + \frac{1}{vl}\sum_i \xi_i - \rho \qquad (2.49)$$

$$sujet\ \text{à}\ \ w.\phi(x_i) \geq \rho - \xi_i, \xi_i \geq 0 \qquad (2.50)$$

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

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

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

$$\frac{1}{2}\sum_{i,j} \alpha_i \alpha_j k(x_i, x_j) \qquad (2.51)$$

$$sujet\ \text{à}\ \ 0 \leq \alpha_i \leq \frac{1}{vl}, i = 1, \dots, l$$

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

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

2.8 Algorithmes d'apprentissage des SVMs

Afin de trouver les paramètres du SVM, il est nécessaire de résoudre le problème d'optimisation quadratique convexe donné par l'équation 2.38 dont la formulation matricielle s'écrit encore:

$$-\frac{1}{2}\alpha^T \widehat{K} \alpha + 1^T \alpha$$

où $\widehat{K}$ est une matrice semi-définie positive dont les composantes $\widehat{K}_{ij} = y_i y_i k(x_i, x_j)$ et 1 est le vecteur unité de même taille que α. Comme la fonction objective est convexe, tout maximum local est aussi un maximum global. Toutefois, il peut y avoir des solutions

optimales différentes en terme de α_i donnant lieu à des performances différentes.

Il existe une grande variété de méthodes et de logiciels traitant de la résolution de problèmes quadratiques[8]. Cependant, quelques méthodes seulement sont capables de traiter un grand nombre d'exemples souvent sous hypothèse que la matrice de Grahm Schmidt K soit creuse **[Veterbei, 1999]**. Dans le cas contraire, l'apprentissage d'un SVM de quelques centaines d'exemples prendrait énormément de temps de calcul et assez de ressources mémoire. Seulement, il est possible de dériver des algorithmes qui exploitent la forme particulière de la fonction objective duale du SVM. Dans cette section, nous allons présenter trois approches différentes pour la résolution du problème quadratique du SVM.

2.8.1 Méthode de chunking

La résolution de la fonction objective duale de l'équation 2.27 avec un très grande nombre d'exemples donne lieu à un vecteur creux. Selon les données, plusieurs des paramètres α_i sont soit nuls ou égales à C. S'il y a moyen de savoir à priori lesquels ai seront nuls, il est possible de réduire la taille de la matrice K sans altérer la valeur de la fonction objective. Aussi, une solution α_i est valide si et seulement si elle respecte les conditions de KKT. Vapnik **[Vapnik, 1982]** était le premier à décrire une méthode qui exploite cette propriété en prenant en compte seulement les α_i non nuls ou ceux violant les conditions de Karush Kuhn Tucker. La taille de ce sous ensemble dépend du nombre de vecteurs de support, de la taille des données et de la complexité du problème de classification. Cette méthode se comporte assez bien sur des problèmes de quelques centaines de vecteurs de support. Des tâches plus complexes requièrent un schéma de décomposition de l'objective en sous problèmes plus facile à résoudre. Cette technique est décrite ci-dessous.

2.8.2 Méthode de décomposition successive

Cette méthode est similaire à celle du «Chunking» dans la mesure où elle considère aussi une succession de sous problèmes quadratiques à résoudre. La différence est que la taille des sous problèmes retenus est fixe. Cette méthode est basée sur la constatation qu'une

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

succession de sous-problèmes quadratiques ayant au moins un exemple qui ne vérifie pas les conditions de KKT converge toujours vers une solution optimale Osuna et al. **[Osuna, et al 1997]** suggèrent de conserver la taille du sous-problème fixe et d'ajouter ou d'enlever un exemple à la fois. Ceci permet d'entraîner de gros ensembles de données. En pratique, cette stratégie peut converger lentement si diverses heuristiques ne sont pas prises en compte. En effet, il est possible d'adopter des stratégies sophistiquées afin d'inclure ou d'exclure quelques exemples de la fonction objective. Différentes stratégies de caching peuvent aussi accélérer la convergence même avec quelques milliers de vecteurs de support. L'algorithme de SVM_{light} est une implémentation de cette méthode **[Joachims, 1999].**

2.8.3 Méthode d'Optimisation par Minimisation Séquentielle: SMO

La méthode d'optimisation par minimisation séquentielle (de l'anglais «Sequential Minimal Optimization») proposée par Platt peut être perçue comme le cas extrême des méthode de décomposition successive **[Platt et al, 1999].** A chaque itération, elle résout un problème quadratique de taille égale à deux. La résolution de ce dernier est analytique et donc nul besoin de recourir à un module d'optimisation quadratique. Encore faut-il choisir le bon couple de variables (α_i, α_j) à optimiser durant chaque itération. Les heuristiques que l'auteur utilise sont basées sur les conditions de KKT. Une autre variante de la méthode a été proposée par Keerthi et al. et ont introduit quelques autres heuristiques à l'algorithme **[Keerthi et al, 1999].** Son implémentation est relativement simple et un pseudo-code de la méthode est aussi fourni **[Platt et al, 1999].** Des versions de l'algorithme adaptés à la régression **[Smola et B. Scholkopf, 1998 , Smola, 1998]** ou aux problèmes mono-classe sont aussi disponibles **[Scholkopf et al, 1999].**

2.8.4 Algorithme de Joachims

L'algorithme de Joachims **[Joachims, 1999]** est une implémentation de SVM_{light}. Il peut traiter un gros ensemble de données allant jusqu'à quelques centaines de milliers d'exemples. Dans cette section, nous décrivons l'algorithme.

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

$$W(\alpha) = \alpha^T 1 + \frac{1}{2}\alpha^T K\alpha \qquad (2.52)$$

$$\text{Sujet à : } \alpha^T y = 0 \qquad (2.53)$$

$$0 \leq \alpha \leq C1 \qquad (2.54)$$

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

Une autre approche possible à cette limitation est de décomposer le problème en une série de tâches moins complexes et plus faciles à résoudre. SVM_{light} est une implémentation modifiée de l'algorithme de décomposition successive de Osuna et al. **[Osuna et al, 1997].** Cette méthode intègre les idées suivantes :

- une stratégie de sélection d'un ensemble actif (sous ensemble des données dont les multiplicateurs α_i sont variables).

-décomposition successive du problème. Ceci exploite les deux propriétés suivantes:

1. il existe beaucoup moins de vecteurs de support que d'exemples d'apprentissage,
2. beaucoup de vecteurs de support ont leurs multiplicateurs α_i égal à C. accélération de la convergence en utilisant la technique du 'caching' et la mise à jour incrémentale des valeurs de a_i.

2.8..4.1 *Stratégie de décomposition*

L'idée s'inspire de la stratégie de décomposition utilisée par Osuna et al. **[Osuna et al, 1997]** qui utilise une stratégie similaire à celle des ensembles actifs **[Gill et al, 1981.].** A chaque itération, les paramètres a_i. sont partagés en deux catégories.

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

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

Tant que les conditions de KKT ne sont pas remplies:

- Sélectionner q paramètres pour l'ensemble actif B. Les variables $l - q$ restantes sont fixées à leurs valeurs courantes.
- Décomposer le problème initial en sous-problèmes QP et optimiser sur l'ensemble actif B.
- Terminer et retourner α.

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

2.8.4.2 *Le problème programmation quadratique*

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

La décomposition de la fonction objective originale $W(\alpha)$ (équation 2.52) est garantie de converger vers une solution optimale si l'ensemble actif B respecte quelques conditions. En particulier, l'ensemble des paramètres variables B est séparé de N. Aussi, nous supposons que α, y et K sont adéquatement arrangés tel que:

$$\alpha = \begin{vmatrix} \alpha_B \\ \alpha_N \end{vmatrix}, y = \begin{vmatrix} y_B \\ y_N \end{vmatrix}, K = \begin{vmatrix} K_{BB} & K_{BN} \\ K_{NB} & K_{NN} \end{vmatrix} \quad (2.55)$$

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

$$\min_\alpha W(\alpha) = -\alpha_B^T(1 - K_{BN}\alpha_N) + \frac{1}{2}\alpha_B^T K_{BB}\alpha_B$$

$$\frac{1}{2}\alpha_N^T K_{NN}\alpha_N - \alpha_N^T 1 \qquad (2.56)$$

Sujet à : $$\alpha_B^T y_B + \alpha_N^T y_N = 0 \qquad (2.57)$$

$$0 \leq \alpha \leq C1 \qquad (2.58)$$

Comme les paramètres dans N sont fixes, les termes $\frac{1}{2}\alpha_N^T K_{NN}\alpha_N$ et $-\alpha_N^T 1$ sont constants et peuvent être omis de la fonction objective. Il en résulte un sous problème beaucoup moins complexe qui peut être résolu avec des packages standards d'optimisation quadratique.

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

2.9 Classification de données multi-classes

Le SVM est un classificateur binaire qui ne traite habituellement que des données appartenant à deux classes. Cependant, il existe des versions plus élaborées prenant en compte plus de deux classes simultanément au sein de la même fonction objective. Il n'est pas prouvé toutefois que celles-ci minimisent le risque structurel. La recherche demeure, cependant, très active sur ce sujet **[Guermeur et Monfrini, 2011].**

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

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

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

2.9.1 Approche Un-contre- Tous

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

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

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

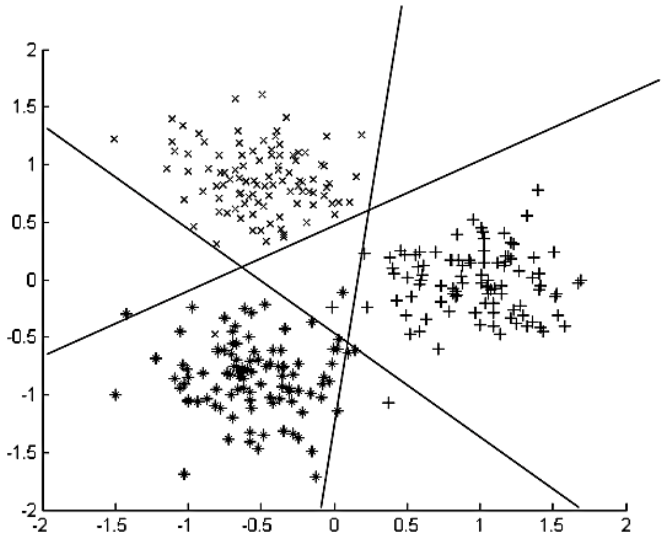


Figure 2. 7. : Problème à trois classes: frontières de décision linéaires dans la stratégie Un-contre-Tous

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

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

Si dans beaucoup de cas, la règle énoncée ci-haut est suffisante, il se peut qu'elle faille lorsque les sorties des K classificateurs ne sont pas comparables. En effet, quelques chercheurs ont mis en évidence ce phénomène pour des données de difficulté moyenne ou grande. Il est facile d'expliquer ceci en considérant le SVM comme étant un perceptron opérant dans un espace défini par l'ensemble de vecteurs de support. D'ailleurs, une analogie intéressante peut être établie entre le SVM et le PMC dans la mesure où l'on sait que la couche de sortie du PMC est un perceptron opérant dans l'espace des neurones cachés. Les poids de sa couche de sortie sont alors équivalents aux paramètres α_i des vecteurs de support. Dans l'approche Un-contre-Tous, les différents SVMs sont entraînés indépendamment les uns des autres. Ils produisent alors des ensembles de vecteurs de support différents. Il en résulte que leurs sorties fournissent des mesures dont les métriques ne sont pas comparables. Nous reportons plus bas les techniques utilisées pour remédier à cet inconvénient.

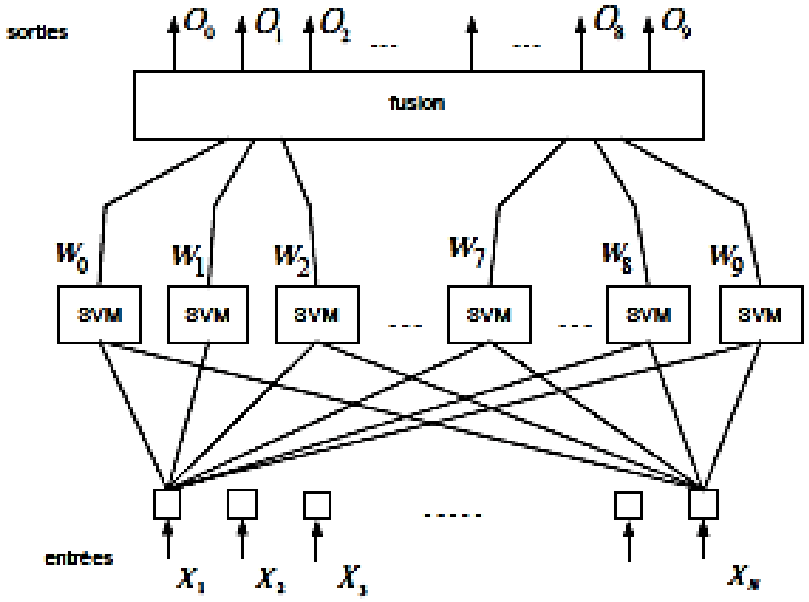


Figure 2. 8.: Architecture du système en stratégie Un-contre-Tous

La figure 2.8 montre l'architecture du système en stratégie Un-Contre-Tous. Notons, que l'étage étiqueté «fusion» désigne le schéma de vote utilisé et aussi toute sorte d'expert capable de fusionner les sorties pour décider de la classe d'appartenance.

2.9.1.1 *Normalisation par la marge*

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

par le facteur de normalisation donné par:

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

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

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

2.9.1.2 *Estimation de la probabilité à posteriori*

Nous allons voir dans cette section qu'il est possible de convertir la sortie brute du SVM en probabilité en utilisant une sigmoïde de la forme:

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

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

2.9.2 Approche Un-contre-Un

L'approche Un-Contre-Un est un cas spécial des méthodes de décomposition proposées par Dietterich. **[Dietterich et Bakiri, 1995]** pour résoudre des problèmes à plusieurs classes. Cette approche requiert la construction de $\frac{1}{2}K(K-1)$ SVM chacun séparant un couple de classes (i, j) parmi ceux existants.

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

$$p_{ij} = P(x \in w_i \big| \boldsymbol{X}, \boldsymbol{X} \in w_i \cup w_j) \qquad (2.65)$$

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

$$\arg\max_{1 \le i \le K} \sum_{j \ne i} [\hat{p}_{ij} > 0.5] \qquad (2.66)$$

L'opérateur $[\eta]$est défini:

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

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

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

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

$$\hat{p}_i = \frac{2}{K(K-1)} \sum_{j \neq 1} \hat{p}_{ij} \qquad (2.67)$$

et la règle de décision s'écrit :

$$\arg\max_{1 \leq i \leq K} \hat{p}_i \qquad (2.68)$$

Nous appellerons cette procédure «reconstruction souple» par opposition à la «reconstruction brute» de l'équation (2.66). Les formulations des équations (2.67) et (2.68) peuvent être généralisées de la façon suivante:

$$\arg\max_{1 \leq i \leq K} \hat{p}_i \, , \hat{p}_i = \frac{2}{K(K-1)} \sum_{j \neq i} \sigma(\hat{p}_{ij}) \, ,$$

où σ est une fonction de seuillage binaire centrée sur 0.5 pour une reconstruction brute, ou bien une fonction identité pour la reconstruction souple.

Ces deux schémas de reconstruction exploitent différemment l'information disponible. Hasti et al. dans **[Hastie et al, 1998.]** démontrent que pour un problème à trois classes et une observation x de test dont la matrice $\boldsymbol{P}$ des probabilités $\hat{p}_{ij}$ est donnée par:

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

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

Aussi, nous avons considéré trois fonctions de reconstruction supplémentaires σ telles que proposées dans **[Moreira et Mayoraz, 1998.]** (cf. tableau 2.2)

Les schémas PWC 4 et PWC 5 permettent de pondérer différemment les réponses positive et négative. La figure 2.10. présente l'architecture simplifiée du système en stratégie Un-contre-Un.

2.9.3 Calcul des probabilités à posteriori des classes

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

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

Tableau 2. 2. : Fonctions d'activation pour la reconstruction

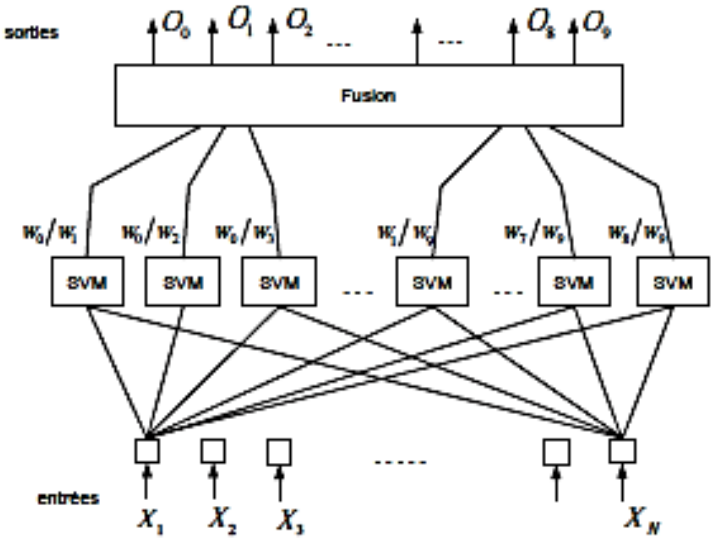


Figure 2. 9. : Architecture du système en stratégie Un-contre-Un

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

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

Cette formule pondère équitablement les estimations $\hat{p}_{ij}$ et ne néglige pas les mesures qui ne sont pas pertinentes. Ceci est particulièrement le cas lorsque x n'appartient à aucune des classes $i\ et\ j$. L'algorithme de Hasti et Tibshirani **[Hastie et Tibshirani, 1998]** a été proposé pour corriger cet inconvénient en utilisant une procédure itérative pour l'estimation des probabilités à posteriori des classes.

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

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

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

Soit les variables auxiliaires:

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

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

$$E = \sum_{i<j} \eta \left[\hat{p}_{ij} log \frac{\hat{p}_{ij}}{\mu_{ij}} + (1 - \hat{p}_{ij}) log \frac{1-\hat{p}_{ij}}{1-\mu_{ij}}\right] \qquad (2.69)$$

Les valeurs de P_i peuvent être estimées en minimisant l'équation (2.69) selon la procédure décrite dans l'algorithme ci-dessous.

Algorithme

---

l) Initialiser $\hat{P}_i$ pour $i = l, \ldots, K$, et calculer (Lij correspondants,

2) répéter jusqu'à convergence :

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

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

c) mise à jour de $\hat{\mu}_{ij}$.

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

Pour des raisons de simplicité, nous considérons $n_{ij} = 1 \; \forall i, j$. Aussi, nous utilisons l'équation (2.69) pour estimer les variables $\hat{\mu}_{ij}$ de départ. Une fois que les probabilités à posteriori $\hat{P}_i(x)$ sont estimées pour tout $i = 1, \ldots, K$, la règle de classification suivante peut être appliquée:

$$c = arg \max_i \left[\hat{P}_i(x)\right] \qquad (2.70)$$

2.9.4 Couplage optimal des classes

Dans l'algorithme de Hastie ci-haut, les coefficients η_{ij} de l'équation (2.69) peuvent s'écrire:

$$N = \begin{bmatrix} - & \eta_{01} & \eta_{02} & \cdots & \eta_{09} \\ \eta_{10} & - & \eta_{12} & \cdots & \eta_{19} \\ \eta_{20} & \eta_{21} & - & \cdots & \eta_{29} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \eta_{90} & \eta_{91} & \eta_{92} & \cdots & - \end{bmatrix} \qquad (2.71)$$

et reflètent l'importance de chaque classificateur (i, j) sur la décision finale étant donné un exemple de test x. Habituellement, ces coefficients sont considérés égaux et sont tous initialisés à un $\forall i, j, tel\ que\ i \neq j$. Donc, tous les classificateurs ont des contributions égales. Par ailleurs, étant donné une donnée de test $x \notin w_i \cup w_i$, l'estimé $\hat{P}_{ij}$ n'est pas pertinent et ne reflète pas une information utile parce que le classificateur (i, j) n'est pas appris sur des exemples de classe autre que $i\ et\ j$. Par conséquent, les utiliser pour estimer les probabilités à posteriori peut biaiser le calcul. En réalité, étant donné une observation de test, pas toutes les sorties $\hat{P}_{ij}$ des SVMs sont informatives, et pertinentes. Certaines d'entre elles peuvent biaiser le résultat surtout pour les exemples ambigus ou difficiles.

Li et al. dans **[Li et al, 2002.]** présentent un algorithme de couplage qui prend en compte ce fait. L'idée est de construire des classificateurs dédiés qui n'utilisent que les valeurs $\hat{P}_{ij}$ pertinentes à la classe qu'on cherche à calculer. En effet, soit la matrice ***P*** des probabilités $\hat{P}_{ij}$ connue. Étant donnée la matrice *N* des coefficients de pondération dans l'équation (2.71), en utilisant l'algorithme de couplage de Hasti et Tibshirani, nous sommes en mesure de construire un classificateur unique. Nous reprenons la définition de [**Li et al, 2002.]** sur le classificateur optimal:

Définition 1 La matrice optimale N^i de coefficients n.u pour la classe w_i; s'écrivant

$$N^i = \{\eta_{kl}\}$$

Vérifie :

$$\begin{cases} \eta_{kl} = 1, si\ k = i\ \ ou\ l = i \\ \eta_{kl} = 0, \qquad\qquad sinon \end{cases}$$

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

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

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

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

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

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

2.9.4.1 *Règle de décision*

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

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

où

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

Dans l'expression précédente, d_k représente une mesure de distance entre le vecteur de probabilités désiré et la sortie du k^{ème} classificateur optimal. Comme mesure de distance,

il est possible d'adopter des métriques différentes telles que l'erreur quadratique donnée par:

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

ou bien l'entropie croisée qui s'écrit:

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

La figure 2.10. montre l'architecture du système avec cette stratégie pour un problème à dix classes (0 à 9).

Conclusion

Ce chapitre est organisé selon trois parties majeures. Après un bref historique sur les SVMs, la première partie introduit la théorie de l'apprentissage statistique et le principe du risque structurel. La notion de régularisation par la maximisation de la marge y est expliquée. Ensuite, nous analysons le principe de la transformation de données dans l'espace augmenté à l'aide des noyaux de Mercer. Plus loin, la formulation mathématique des SVMs y est présentée.

Dans la deuxième partie nous discutons des stratégies d'apprentissage des SVMs sur de gros ensembles de données. Enfin, dans la dernière partie, nous abordons les différentes stratégies de combinaison pour la classification de données multi classes. En particulier, nous expliquons l'approche un-contre-un et discutons les différents schémas de vote afin de fournir une classe d'appartenance.

Chapitre III :

Techniques pour la Réduction de Dimension

3.1 Introduction

La taille des données peut être mesurée selon deux dimensions, le nombre de variables et le nombre d'exemples. Ces deux dimensions peuvent prendre des valeurs très élevées, ce qui peut poser un problème lors de l'exploration et l'analyse de ces données. Pour cela, il est fondamental de mettre en place des outils de traitement de données permettant une meilleure compréhension de la valeur des connaissances disponibles dans ces données. La réduction des dimensions est l'une des plus vieilles approches permettant d'apporter des éléments de réponse à ce problème. Son objectif est de sélectionner ou d'extraire un sous-ensemble optimal de caractéristiques pertinentes pour un critère fixé auparavant. La sélection de ce sous-ensemble de caractéristiques permet d'éliminer les informations non-pertinentes et redondantes selon le critère utilisé. Cette sélection/extraction permet donc de réduire la dimension de l'espace des exemples et de rendre l'ensemble des données plus représentatif du problème. En effet, les principaux objectifs de la réduction de dimension sont :

- faciliter la visualisation et la compréhension des données,
- réduire l'espace de stockage nécessaire,
- réduire le temps d'apprentissage et d'utilisation,
- identifier les facteurs pertinents.

Les algorithmes d'apprentissage artificiel requièrent typiquement peu de traits - features - ou de variables - attributs - très significatifs caractérisant le processus étudié. Dans le domaine de la reconnaissance des formes et de la fouille de données, il pourrait encore être bénéfique d'incorporer un module de réduction de la dimension dans le système global avec comme objectif d'enlever toute information inconséquente et redondante. Cela a un effet important sur la performance du système. En effet le nombre de caractéristiques utilisées est directement lié à l'erreur finale. L'importance de chaque caractéristique dépend de la taille de la base d'apprentissage - pour un échantillon de petite taille, l'élimination d'une caractéristique importante peut diminuer l'erreur. Il faut aussi noter que des caractéristiques individuellement peu pertinentes peuvent être très informatives si on les utilise conjointement. La réduction de la dimension est un problème complexe qui permet de réduire le volume d'informations à traiter et de

faciliter le processus de l'apprentissage. Nous pouvons classer toutes les techniques mathématiques de réduction de dimension en deux grandes catégories :

- la sélection de variables : qui consiste à choisir des caractéristiques dans l'espace de mesure.
- l'extraction de traits : qui vise à sélectionner des caractéristiques dans un espace transformé - dans un espace de projection

Nous nous limitons dans cette thèse à l'étude des approches de réduction par extraction de traits.

Dans ce chapitre, nous considérons un ensemble de données composé de n individus. Chaque individu est représenté par un vecteur dans l'espace de m dimensions. Chaque vecteur est la réalisation de m variables aléatoires $\frac{(X_1,X_2,\dots X_m)}{X:\, m \text{ x } n}$. L'ensemble des données est représenté sous la forme d'une matrice contenant en ligne les dimensions et en colonne les individus :

$$X = \{x_{j,i} \ ,1 \leq j \leq m, 1 \leq i \leq n\}$$

où n est le nombre d'éléments et m est le nombre de dimensions de chaque élément.

Définition 3.1 (Réduction du monde de dimensions) :

Etant donné un vecteur colonne m-dimensionnel $\vec{x}_i = (x_{1,i},\ x_{2,i}, \dots, x_{m,i},)^T$ il s'agit de trouver une autre représentation de dimension réduite $\vec{y}_i = (y_{1,i},\ y_{2,i}, \dots, y_{m,i},)^T$ avec $r \leq m$, qui exprime la même information que les données originales. Les composantes de $\vec{y}_i$ son appelées composantes cachées, variable latentes ou dimensions intrinsèques.

Définition 3.2 (Techniques linéaires de réduction de dimensions)

Les techniques linéaires de réduction de dimensions permettent de transformer les données originales dans un espace de dimension plus réduite tel que la nouvelle représentation des données soit une combinaison linéaire des données originales, c'est-à-dire :

$y_{ki} = w_{k1}.x_{1i} + w_{k1}.x_{1i} + \cdots + w_{k1}.x_{1i}\ \forall k \in \{1,2, \dots r\}, \forall i \in \{1,2, \dots . n\}$

ou, en notation matricielle :

$$Y = \Omega \mathrm{X}$$

avec $X = \{x_{j,i}, 1 \leq j \leq m, 1 \leq i \leq n\}, Y = \{y_{k,i}, 1 \leq k \leq r, 1 \leq i \leq n\}$

et $\Omega = \{w_{k;j}, 1 \leq k \leq r, 1 \leq j \leq m\}$ la matrice de transformation linéaire.

La transformation inverse , est :

$x_{j,i} = a_{j,1}.y_{1,i} + a_{j,2}.y_{2,i} + \cdots + a_{j,r}.y_{r,i} \quad \forall j \in \{1,2,\ldots,m\}, \forall i \in \{1,2,\ldots,n\}$

ou en notation matricielle :

$$X = AY$$

avec $A = \{a_{jk}, 1 \leq j \leq m, 1 \leq k \leq r\}$ la matrice de transformation inverse.

Toutes les techniques d'analyse linéaire sont basées sur le même principe, elles se distinguent par la manière dont sont déterminées les matrices de transformation Ω et A. Après avoir décrit le problème de la malédiction de la dimension, nous présentons dans cette partie deux techniques d'analyse en composantes : l'analyse en composantes principales (ACP) et l'analyse en composantes indépendantes (ACI), ainsi qu'un critère d'analyse discriminante : le critère de Fisher utilisé dans l'analyse linéaire discriminante (LDA).

3.2 Propriétés des espaces à grandes e dimension

$\boldsymbol{m}$	2	3	10	16	20	40	50	100
Nombre de coins	4	8	1024	65536	$\simeq \mathbf{10^6}$	$\simeq \mathbf{10^{12}}$	$\simeq \mathbf{10^{15}}$	$\simeq \mathbf{10^{30}}$

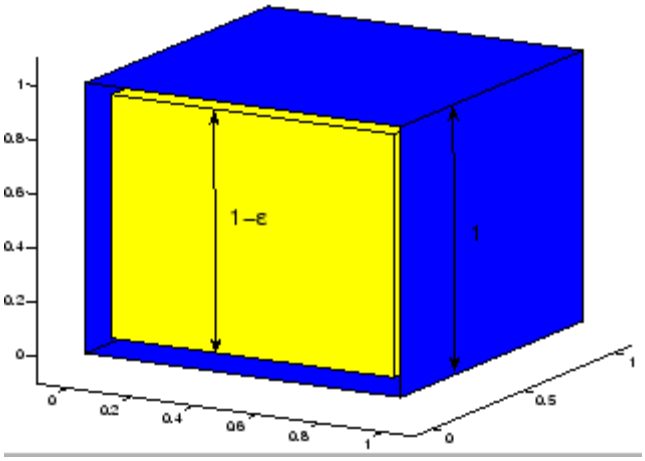
Tableau 3. 1. : Nombre de coins d'un hypercube en fonction du nombre de dimensions $\boldsymbol{m}$

L'apprentissage et la recherche par similarité sur des données de grande dimensions (plus de 16 dimensions d'après **[Weber *et al.*, 1998]** sont des problèmes difficiles en partie à cause du problème de la malédiction de la dimension (appelé aussi fléau de la dimensionnalité ou *curse of dimensionality* **[Bellman, 1961].** En effet, dans les espaces de grande dimension, les intuitions géométriques valables en faible dimension peuvent se révéler fausses. C'est pourquoi certains algorithmes efficaces en faible dimension vont se révéler peu performants en grande dimension.

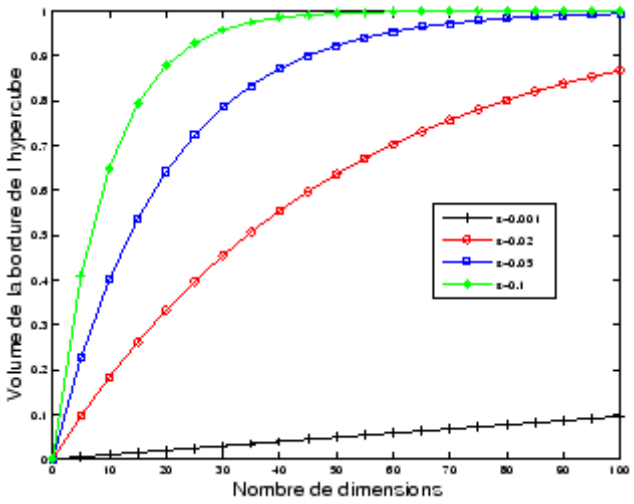
Pour comprendre ce problème, on peut étudier le nombre de coins d'un hypercube de dimension m (voir tableau 3.1.). Le nombre de coins croit de manière exponentielle en fonction du nombre de dimensions. En dimension 100, le nombre de coins est beaucoup plus grand que le nombre de données d'apprentissage. Or dans les bases de données d'images, il est fréquent que le nombre de dimensions soit plus important que 100, et le nombre de valeurs que peut prendre une dimension est souvent supérieur à 2. Le nombre d'exemples est donc petit par rapport à la taille de l'espace. Si on partitionne l'espace en séparant en deux chaque dimension, le nombre de *clusters* formés est exponentiel en fonction du nombre de dimensions. De plus, plus le nombre de dimensions est grand, plus il y a de *clusters* qui ne contiennent aucune donnée. On obtient donc l'effet contraire à l'objectif du partitionnement (réduire le nombre d'individus en les regroupant au sein de *clusters*).

Dans **[Beyer *et al.*, 1999],** Beyer *et al.* démontrent que plus le nombre de dimensions est grand, plus le rapport entre la distance minimale et la distance maximale entre les données tend vers un, autrement dit plus la dimension des données augmente, plus les vecteurs ont tendance à être équidistants, la notion du plus proche voisin n'existe plus.

Un autre phénomène des espaces à grande dimension concerne les formes géométriques. Prenons un hypercube dont tous les côtés sont de longueur 1 unité. La bordure de cet hypercube est la zone située entre le sous-hypercube de coté de longueur $1-\in$ et l'hypercube lui-même (cf. figure 3.1. (a)). Son volume vaut $1-(1-\in)^m$. Ainsi pour $\in= 5\%$, et $m = 40$, le volume de la bordure vaut 87.1% du volume de l'hypercube ! Le volume de la bordure tend vers 1 et le nombre de dimensions devient grand, autrement dit la bordure occupe presque tout le volume de l'hypercube (cf. figure 3.1.(b)) **[Hérault et al, 2002].** Ce phénomène implique que lorsque l'on ajoute à un *cluster*, représenté par un hypercube, un vecteur qui semble proche de la bordure de cet hypercube pour former un nouvel hypercube englobant le vecteur, le nouvel hypercube a un volume considérablement plus grand que l'ancien hypercube. Il implique aussi que la probabilité qu'un vecteur se trouve près d'une bordure est très grande.



(a) la bordure d'un hypercube est la zone située entre l'hypercube de coté 1 unité et l'hypercube de coté $1 - \epsilon$ unités



(b) Volume de la bordure d'un hypercube en fonction du nombre de dimensions pour différentes valeurs de ϵ

Figure 3. 1. : Propriété non intuitive de la bordure d'un hypercube en grande dimension

Pour résumer, citons les 5 difficultés des recherches par similarité énumérées par [**Weber et al, 1998]** dans les espaces à grande dimension :

- le partitionnement exponentiel de l'espace,
- la faible occupation de l'espace par les données,

- le rétrécissement du volume des hypersphères centrées sur les requêtes contenant les plus proches voisins,
- l'augmentation exponentielle de la taille des bases de données,
- l'augmentation de la dispersion des données.

Le lecteur intéressé par d'autres exemples de problèmes en grande dimension pourra se référer à l'excellent état de l'art de S.-A. Berrani **[Berrani, 2004].**

3.3 Techniques linéaires de Réduction de dimension.

3.3.1 Analyse en Composantes Principales (ACP)

La plus célèbre méthode de réduction du nombre de dimensions est l'Analyse en Composantes Principales (ACP). Cette analyse non-supervisée recherche les axes (facteurs principaux) qui permettent une dispersion maximale des données, puis projette les données dans le nouvel espace réduit obtenu. Elle trouve ses origines dans les travaux de **[Hotelling, 1933].** Elle est aussi appelée décomposition en valeurs singulières (SVD), transformée de Karhunen-Loève (KLT), transformée de Hotelling ou méthode de fonction orthogonale empirique, en fonction du domaine d'application. L'ACP est la transformation linéaire optimale pour garder le sous-espace qui a la plus grande variance. Cependant, elle nécessite plus de calcul que d'autres méthodes comme la transformée en cosinus discret (DCT) notamment. A la différence de la plupart des autres transformées linéaires, l'ACP n'a pas un ensemble fixe de composantes principales : elles sont dépendantes des données. L'ACP peut être utilisée pour :

- réduire le nombre de dimensions,
- compresser des données,
- filtrer du bruit,
- identifier par corrélation.

La réduction du nombre de dimensions implique une perte d'information. On veut cependant trouver le sous-espace qui permet de perdre le moins d'information. **[Saporta,**

1990] a montré que la recherche des axes du sous-espace peut se faire de manière séquentielle. On commence par rechercher l'axe qui décrit le mieux les données, puis le deuxième qui de plus doit être orthogonal au premier, et ainsi de suite. Soit $\vec{x}_i$ un point de l'espace multidimensionnel initial, et $\vec{y}_i$ le point de l'espace initial qui est son projeté orthogonal dans le sous-espace réduit, l'information perdue par la projection peut être mesurée par la distance euclidienne $\|\vec{y}_i - \vec{x}_i\|$ entre le point $\vec{x}_i$ et le point $\vec{y}_i$ (tous les deux représentés dans l'espace initial). On cherche alors la transformation linéaire qui minimise la distance entre les points $\vec{x}_i$ et leur projeté $\vec{y}_i$. Cela revient à trouver le sous-espace tel que la moyenne des carrés des distances (dispersion) entre les points et leur projection :

$$j_{ACP} = \sum_{i=1}^{n} \|\vec{y}_i - \vec{x}_i\|^2 \qquad (3.1)$$

soit minimale. Cherchons d'abord le point de l'espace tel que la projection des points de l'espace sur cet unique point permette une perte d'information minimale. Soit $\vec{\mu} = (\mu_1, \mu_2, \dots \mu_m,)$ le vecteur moyen. On peut réécrire le critère j_{ACP} sous la forme :

$$j_{ACP} = \sum_{i=1}^{n} \|(\vec{y}_i - \vec{x}_i) - (\vec{x}_i - \vec{\mu})\|^2$$

$$= \sum_{i=1}^{n} \|(\vec{y}_i - \vec{\mu})\|^2 + \sum_{i=1}^{n} \|(\vec{x}_i - \vec{\mu})\|^2 \qquad (3.2)$$

Le critère j_{ACP} est minimal quet $\vec{y}_i = \vec{\mu}$, c'est-à-dire que tous les points sont projetés sur le centre de gravité. Le centre du sous-espace qui minimise j_{ACP} est le centre de gravité $\vec{\mu}$.

Pour minimiser le critère j_{ACP} , on cherche la droite passant par le centre de gravité $\vec{\mu}$ et de vecteur unitaire $\vec{u}$, qui minimise les distances $\|(\vec{y}_i - \vec{x}_i)\|$ entre les projetés $\vec{y}_i$ de $\vec{x}_i$ sur cette droite et les points $\vec{x}_i$. On peut réécrire les projetés sous la forme :

$$\vec{y}_i = \vec{\mu} + (\vec{y}_i - \vec{\mu}) = \vec{\mu} + \Upsilon_i u \qquad (3.3)$$

Où $\Upsilon_i = \|\vec{y}_i = \vec{\mu}\|$ est la distance entre le point $\vec{y}_i$ et le centre de gravité $\vec{\mu}$. Minimiser j_{ACP} revient à trouver la droite qui minimise les coefficients Υ_i .

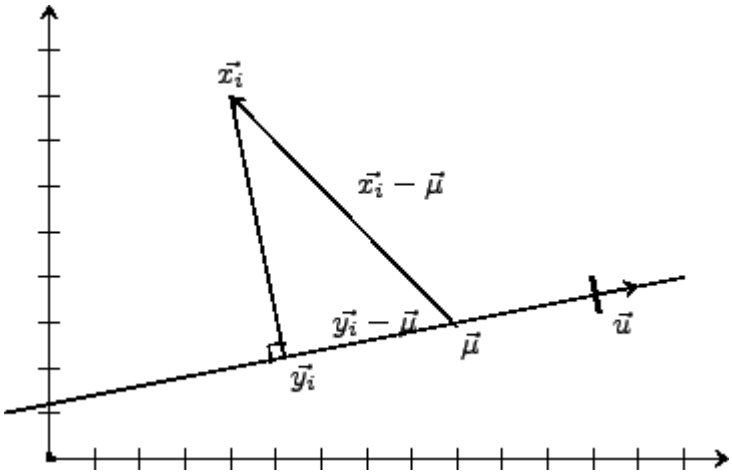


Figure 3. 2. : Projection orthogonale du point $\vec{x_i}$ sur une droite passant par le centre de gravité $\vec{\mu}$.

Si l'on remplace $\vec{y}_i$ par $\vec{\mu} + \Upsilon_i \vec{u}$ dans l'équation (3.12), on obtient :

$$j_{ACP} = \sum_{i=1}^{n} \|(\vec{\mu} + \Upsilon_i \vec{u}) - \vec{x}_i\|^2$$

$$= \sum_{i=1}^{n} \|\Upsilon_i \vec{u} - (\vec{x}_i + \vec{\mu})\|^2$$

$$= \sum_{i=1}^{n} \Upsilon_i^2 \|\vec{u}\|^2 - 2\sum_{i=1}^{n} \Upsilon_i \vec{u}^T(\vec{x}_i - \vec{\mu}) + \sum_{i=1}^{n} \|\vec{x}_i - \vec{\mu}\|^2 \qquad (3.4)$$

$$= \sum_{i=1}^{n} \Upsilon_i^2 - 2\sum_{i=1}^{n} \Upsilon_i \vec{u}^T(\vec{x}_i - \vec{\mu}) + \sum_{i=1}^{n} \|\vec{x}_i - \vec{\mu}\|^2$$

Or $\sum_{i=1}^{n} \Upsilon_i^2 - 2\sum_{i=1}^{n} \Upsilon_i \vec{u}^T(\vec{x}_i - \vec{\mu})$ vaut zéro quet :

$$\Upsilon_i = \vec{u}^T(\vec{x}_i - \vec{\mu}) \qquad (3.5)$$

Géométriquement, ce résultat signifie que nous obtenons une solution au sens des moindres carrés en projetant $\vec{x}_i$ sur la droite de direction $\vec{u}$ passant par $\vec{\mu}$. Soit $S = \sum_{i=1}^{n} (\vec{x}_i - \vec{\mu})(\vec{x}_i - \vec{\mu})^T$ la matrice de dispersion (*scatter matrix*), si l'on remplace Υ_i par $\vec{u}^T(\vec{x}_i - \vec{\mu})$ dans l'équation (3.5), on obtient :

$$j_{ACP} = \sum_{i=1}^{n} [\vec{u}^T(\vec{x}_i - \vec{\mu})]^2 + \sum_{i=1}^{n} \|\vec{x}_i - \vec{\mu}\|^2$$

$$= \sum_{i=1}^{n} \vec{u}^T (\vec{x}_i - \vec{\mu})(\vec{x}_i - \vec{\mu})^T \vec{u} + \sum_{i=1}^{n} \|\vec{x}_i - \vec{\mu}\|^2$$

$$= \vec{u}^T S\vec{u} + \sum_{i=1}^{n} \|\vec{x}_i - \vec{\mu}\|^2 \qquad (3.6)$$

Minimiser le critère j_{ACP} revient donc à maximiser le critère j'_{ACP} tel que :

$j'_{ACP} = \vec{u}^T S\vec{u}$ (26)

On cherche donc $\vec{u}_1$ tel que :

$$\vec{u}_1 = \max_{\vec{u}} \vec{u}^T S\vec{u} \qquad (3.7)$$

La solution de ce problème de maximisation d'une forme quadratique est le vecteur propre de S associé à la plus grande valeur propre λ_1 . On cherche alors le vecteur $\vec{u}_2$ orthogonal à $\vec{u}_1$ tel que la dispersion des points projetés sur cet axe soit maximale. On montre que $\vec{u}_2$ est le vecteur propre de S associé à la deuxième plus grande valeur propre. Plus généralement, le sous-espace à r dimensions recherché est engendré par les r premiers vecteurs propres de la matrice de covariance S des variables aléatoires $(X_1, X_2, \dots, X_m)$ associés aux plus grandes valeurs propres. Les vecteurs $\vec{u}_j$ sont appelés *facteurs principaux*. Les *composantes principales* sont les variables artificielles définies par les n valeurs résultantes de la projection de l'ensemble des vecteurs originaux sur un axe principal.

Nous donnons maintenant un mode opératoire pour réaliser une ACP. Pour trouver les principales composantes, on peut réaliser les opérations suivantes :

1. organiser les données en vecteurs colonnes pour obtenir une matrice $X: m \text{ x } n$,
2. calculer le vecteur moyen de chaque dimension $\vec{\mu} = (\mu_1, \mu_2, \dots, \mu_m)^T$,
3. soustraire le vecteur moyen à toutes les colonnes de X afin d'obtenir des données centrées, la matrice obtenue est appelée M,
4. calculer la matrice de covariance $S: m \text{ x } m$ de $(S = M.M^T)$,
5. calculer et trier par ordre décroissant des valeurs propres λ_j , les vecteurs propres $\vec{\mu}_j$ de S , ils forment les colonnes de la matrice $U: m \text{ x } m$,
6. garder les r premières colonnes de U pour former la matrice $\widetilde{U}: m \text{ x } r$.

La variance dépend de l'ordre de grandeur des descripteurs, lorsque les descripteurs sont de types différents, les données X doivent être centrées réduites. La matrice de covariance devient alors une matrice de corrélation.

Pour calculer les coordonnées du projeté $\vec{y}_i$ de $\vec{x}_i$ dans l'espace réduit, on réalise l'opération :

$$\vec{y}_i = \widetilde{U}^T(\vec{x}_i - \vec{\mu}) = \widetilde{U}^T \vec{x}_i{}^c \qquad (3.8)$$

où $\vec{x}_i{}^c$ est le vecteur dans l'espace centré de $\vec{x}_i$. Nous obtenons donc bien une transformation linéaire.

Une des difficultés de l'ACP est le choix du nombre de dimensions de l'espace réduit. Le pourcentage d'inertie totale expliquée défini par :

$$\frac{\sum_{k=1}^{r} \lambda_k}{\sum_{j=1}^{m} \lambda_j} \qquad (3.9)$$

donne la proportion cumulée de la variance exprimée par les r premières composantes. En se basant sur ce pourcentage l'utilisateur peut choisir le nombre de composantes à retenir *à posteriori* en fonction de la quantité d'information qu'il souhaite conserver. Une autre méthode est le *critère de Kaiser* à réaliser sur des données centrées réduites, il consiste à ne retenir que les composantes principales correspondant à des valeurs propres supérieures à 1, ce qui revient à ne retenir que les composantes qui apportent plus que les variables initiales. Une troisième méthode est la *règle du coude* qui consiste à détecter un coude sur le diagramme des valeurs propres. Le choix du nombre d'axes à retenir n'a pas de solutions rigoureuses.

3.3.2 Analyse en Composantes Indépendantes (ACI)

Alors que l'ACP est une méthode qui recherche les axes (décorrélés et orthogonaux) qui représentent le mieux les données, l'analyse en composantes indépendantes (ACI) est une méthode non-supervisée qui recherche les axes qui sont les plus statistiquement indépendants les uns des autres (et donc décorrélés, mais pas forcément orthogonaux). Habituellement en traitement du signal, on suppose que les signaux sont seulement décorrélés, l'indépendance est une notion plus forte que la décorrélation. Dans le cas de variables gaussiennes, l'indépendance et la décorrélation sont équivalentes.

Historiquement, l'ACI a été introduite par Jeanny Herault et Christian Jutten en 1985 **[Jutten, 1987]**. Utilisée initialement pour des problèmes de séparation aveugle de sources, elle peut être également utilisée pour l'analyse et la compression de données, la détection bayésienne, la localisation de sources, et l'identification et la déconvolution aveugle. L'ACI n'est pas un outil de réduction du nombre de dimensions à proprement parlé, car dans certaines variantes de l'ACI, le nombre de dimensions de l'espace de projection est plus grand que le nombre de dimensions de l'espace initial. Elle est d'ailleurs, dans la pratique, souvent utilisée conjointement à une ACP. Mais nous nous intéressons à cette technique, car en plus de permettre la réduction du nombre de dimensions, elle a été utilisée récemment pour la classification de scènes naturelles **[Le Borgne et al, 2003]**. En effet, les neurophysiologistes se sont intéressés aux représentations efficaces des images et en particulier aux images de scènes naturelles. Leur théorie est la suivante : si les cellules du cortex visuel primaire sont affectées à des tâches spécifiques de reconnaissance de certaines structures géométriques, c'est peut-être parce que cette solution biologique au problème de la lecture d'une image est optimale en terme de l'analyse et de compression des données. Pour accéder efficacement à l'information pertinente, il faut optimiser le contraste, c'est-à-dire disposer de plusieurs points de vue. Par exemple, pour étudier un objet tridimensionnel, il vaut mieux prendre trois angles de vue les plus différents possible que trois angles de vue proches. L'ACI a également été utilisée pour analyser conjointement le texte et l'image **[Kolenda et al, 2002]**.

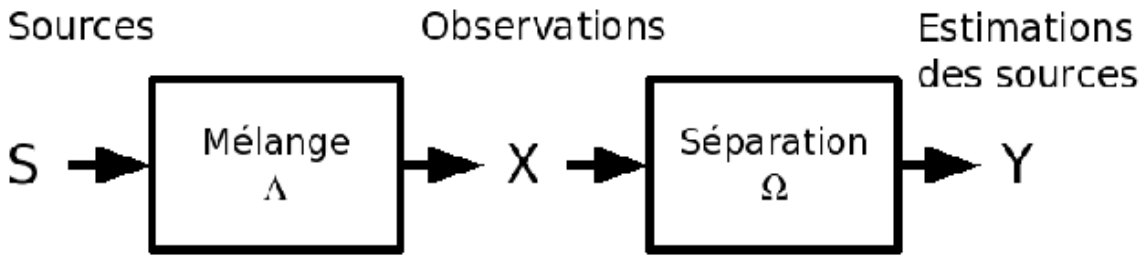


Figure 3.3. : Principe de l'ACI

- **Principe**

On suppose que les données ont été générées par r sources distinctes :

$$x_{j,i} = \alpha_{j,1}.s_{1,i} + \alpha_{j,2}.s_{2,i} + \cdots.\alpha_{j,r}.s_{r,i} \quad \forall j \in \{1,2,\dots,m\}, \forall i \in \{1,2,\dots n\} \qquad (3.10)$$

ou en notation matricielle :

$$X = AS \quad (3.11)$$

On cherche alors la matrice Ω telle que :

$Y = \Omega X = \Omega AS \; et \; Y \; est \; une \; bonne \; estimation \; de \; S$ (3.12)

La figure 3.3. résume le principe de l'ACI.

Par construction, les sources $S_1, S_2, \dots S_r$ sont indépendantes, c'est pourquoi on souhaite que les composantes $Y_1, Y_2, \dots Y_r$ soient les plus indépendantes possibles. Rappelons d'abord que l'indépendance statistique nécessite que la densité de probabilité conjointe soit le produit des probabilités des lois marginales, c'est-à-dire :

$$P(Y_1, Y_2, \dots Y_r) = P(Y_1)\, P(Y_2) \dots . P(Y_r) \quad (3.13)$$

Soit $\Phi(Y_1, Y_2, \dots, Y_r)$ une mesure de l'indépendance des composantes $Y_1, Y_2, \dots, Y_r$, trouver les composantes $Y_1, Y_2, \dots, Y_r$ les plus indépendantes revient à trouver la matrice Ω tel que :

$$\min \Phi(\Omega X) \quad (3.14)$$

Il existe un grand nombre de méthodes pour mesurer l'indépendance, la méthode choisie détermine le principe de l'algorithme d'ACI. Il existe deux principales familles de mesures d'indépendance : celle basée sur le maximum de vraisemblance, et celle basée sur la non-gaussianité.

Posons $\vec{y}_i = \left(y_{1,i}, y_{2,i}\ , \dots, y_{r,i}\right)^T = \Omega \vec{x}_i$. Une première méthode, appartenant à la première famille, définie $\Phi(\vec{y}_i)$ tel que :

$$\Phi_{IL}(\vec{y}_i) = div\ KL\ (P_{\vec{y}_i}, \prod_{k=1}^{r} P_{y_{k,i}}) \quad (3.15)$$

où $P_{\vec{y}_i} (= P_{(y_{1,i},\ y_{2,i}, \dots, y_{r,i})})$ est la densité de probabilité (ou distribution) de $\vec{y}_i, P_{y_{k,i}}$ est la densité de probabilité de Y_k et $divKL$ est la divergence de Kullback-Leibler. Une autre méthode, basée sur la non-gaussianité, c'est-à-dire l'écart entre la distribution de $\vec{y}_i$ et la distribution gaussienne de mêmes moyenne $\mu_{\vec{y}_i}$ et écart-type $\sigma_{\vec{y}_i}$, définie par :

$$\Phi_G(\vec{y}_i) = divKL(P_{\vec{y}_i}, N(\mu_{\vec{y}_i}, \sigma_{\vec{y}_i})) \qquad (3.16)$$

Trois conditions sont requises pour l'estimation du modèle ACI :

- au plus une composante peut être gaussienne,
- la matrice de mélange doit être de rang plein,
- le nombre de vecteurs d'attributs doit être supérieur au nombre de composantes à estimer (sources). Le nombre de composantes indépendantes correspond souvent au nombre de dimensions des vecteurs attributs originaux.

Dans la pratique, l'ACI nécessite que les données soient centrées et on utilise une ACP avant l'ACI pour obtenir une matrice diagonale. L'ACP transforme les données dans un espace réduit et décorrèle les données, puis l'ACI transforme les données de l'espace réduit vers un nouvel espace dans lesquels les dimensions sont indépendantes.

3.3.3 Analyse Linéaire Discriminante (LDA)

Nous venons de voir que l'ACP et l'ACI peuvent être utilisées pour réduire le nombre de dimensions. Cependant, ce sont des techniques non-supervisées, elles n'utilisent donc pas les classes associées aux données. Contrairement à l'ACP, l'objectif de l'Analyse Linéaire Discriminante (LDA) est de réduire le nombre de dimensions tout en préservant au maximum les classes. Pour cela, elle cherche les axes tels que la projection des données dans l'espace engendré par ces axes permette une plus grande séparation des classes. La figure 3.7. compare les axes choisis par la LDA et l'ACP pour les mêmes données.

Nous donnons, dans une première partie, le principe de l'analyse de Fisher dans le cas de la séparation en deux classes par une seule droite. Puis, dans la deuxième partie, nous résumons les formules de la LDA dans le cas général.

3.3.3.1 *Cas à deux classes*

Supposons que nous disposons d'un ensemble d'apprentissage où chaque document appartient à l'une des deux classes C_1 ou C_2 (k =1 ou k =2). Nous cherchons l'axe de direction $\vec{\omega} = (\omega_1, \omega_2, \dots, \omega_m)^T$ tel que les projetés des éléments soient le mieux séparés en classes. Soit le scalaire y_i représentant le projeté de $\vec{x}_i$ sur cet axe, on a :

$$y_i = \vec{\omega}^T . \vec{x}_i \quad (3.17)$$

Une mesure de la séparation entre les points projetés est la différence des vecteurs moyens de chaque classe. Le vecteur moyen de la classe C_k est : $\mu = \frac{1}{nC_k} \sum_{\vec{x}_i \in C_k} \vec{x}_i$

Le vecteur moyen de la classe C_k dans l'espace engendré par $\vec{w}^T$ est le scalaire :

$$\mu'_k = \frac{1}{nC_k} \sum_{y_i \in C_k} y_i = \frac{1}{nC_k} \sum_{\vec{x}_i \in C_k} \vec{\omega}^T . \vec{x}_i = \vec{\omega}^T . \vec{\mu}_k \quad (3.18)$$

C'est le projeté du point $\vec{\mu}_k$ sur $\vec{\omega}$. La distance au carré entre les projetés des vecteurs moyens des classes est :

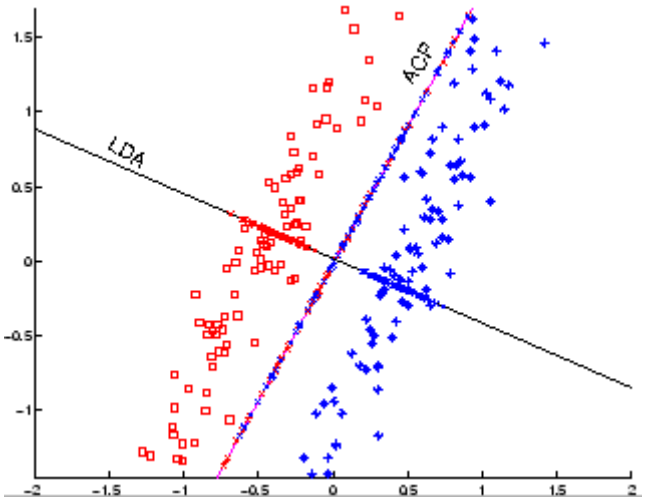
$$\left|\mu'_1 - \mu'_2\right|^2 = (\vec{\omega}^T . \vec{\mu}_1 - \vec{\omega}^T . \vec{\mu}_2)^2 = \vec{\omega}^T (\vec{\mu}_1 - \vec{\mu}_2)(\vec{\mu}_1 - \vec{\mu}_2)^T \vec{\omega} = \vec{\omega}^T S_B \vec{\omega} \quad (3.19)$$

où $S_B = (\vec{\mu}_1 - \vec{\mu}_2)(\vec{\mu}_1 - \vec{\mu}_2)^T$ est la matrice de dispersion interclasse. $S_B \vec{\omega}$ est toujours dans la direction de $\vec{\mu}_1 - \vec{\mu}_2$. Pour séparer au mieux les projetés en classes, la valeur de $\left|\mu'_1 - \mu'_2\right|^2$ doit être grande par rapport à la dispersion des projetés de chaque classe C_k , définie par :

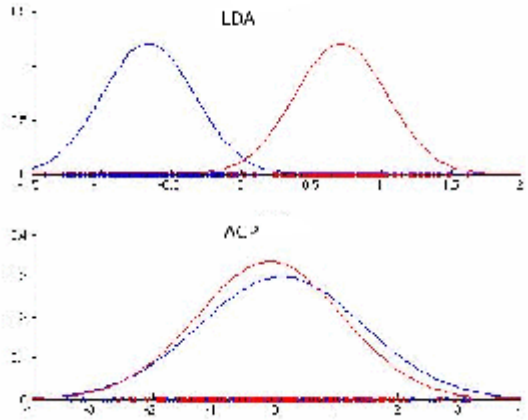
$$S'^2_k = \sum_{y_i \in C_k} (y_i - \mu'_k)^2 = \sum_{\vec{x}_i \in C_k} (\vec{\omega}^T . \vec{x}_i - \vec{\omega}^T . \vec{u}_k)^2 = \sum_{\vec{x}_i \in C_k} \vec{\omega}^T (\vec{x}_i - \vec{u}_k)(\vec{x}_i - \vec{u}_k)^T \vec{\omega}$$

$$= \vec{\omega}^T S_k \vec{\omega} \quad (3.20)$$

Avec S'^2_k la dispersion (proportionnelle au carré de l'écart-type) de la classe C_k et S_k la matrice de dispersion de la classe C_k . On appelle matrice de dispersion intraclasse la matrice $S_\omega = S_1 + S_2$.



(a) Un ensemble de données séparées en deux classes distinctes. Les axes fournis par la LDA et l'ACP



(b) La projection des données sur les axes montre que la LDA sépare les classes (en haut), tet is que l'ACP les confond (en bas).

Figure 3. 4. : Exemple de réduction de dimensions par LDA et par ACP. Les deux méthodes permettent de réduire de 2 à 1 dimension en trouvant, soit l'axe qui décrit le mieux les données pour l'ACP, soit l'axe qui sépare le mieux les classes pour la LDA.

Finalement, le critère de Fisher qui vise à mesurer la séparation des classes dans l'espace de projection engendré par $\vec{\omega}$ est défini par :

$$J_{LDA}(\vec{\omega}) = \frac{\left|\mu'_1 - \mu^*_2\right|^2}{S'^2_1 + S'^2_2} = \frac{\vec{\omega}^T S_B \vec{\omega}}{\vec{\omega}^T S_\omega \vec{\omega}} \qquad (3.21)$$

Remarquons que ce critère suppose que les données de chaque classe sont réparties dans l'espace de projection sous la forme d'une gaussienne de vecteur moyen μ'_k et d'écart-type σ'_k (proportionnel à S'_k).

On cherche $\vec{\omega}$ qui maximise le critère de Fisher J_{LDA} . On peut exprimer ce problème sous la forme d'un problème de valeurs propres $S_B\vec{\omega} = \lambda\vec{\omega}$. Si S_W n'est pas singulière, on montre que dans notre cas à deux classes, les solutions sont de la forme :

$$\vec{\omega} = S_W^{-1}(\vec{\mu}_1 - \vec{\mu}_2) \qquad (3.22)$$

On peut utiliser la LDA pour construire un classificateur. Pour cela, il suffit en plus de trouver la borne qui sépare les $\vec{x}_i$ de C_1 et ceux de C_2 .

3.3.3.2 *Cas général*

Dans le cas général, la matrice de dispersion totale (*total scatter matrix*) est définie par :

$$S_T = \sum_{\vec{x}_i \in \mathrm{A}} (\vec{x}_i - \vec{\mu}_i)(\vec{x}_i - \vec{\mu})^T \qquad (3.23)$$

où $\vec{\mu}$ est le vecteur moyen des données. La matrice de dispersion intraclasse (*within-class scatter matrix*) est définie comme la dispersion des moyennes de chaque classe :

$$S_W = \sum_{k=1}^{n_C} \sum_{\vec{x}_i \in C_k} (\vec{x}_i - \vec{\mu}_k)(\vec{x}_i - \vec{\mu}_k)^T = \sum_{k=1}^{n_C} S_k \qquad (3.24)$$

où S_k est la matrice de dispersion des éléments de la classe C_k . La matrice de dispersion interclasse (*between-class scatter matrix*) est définie comme la moyenne des dispersions de chaque classe :

$$S_B = \sum_{k=1}^{n_C} n_{C_k}(\vec{\mu}_k - \vec{\mu})(\vec{\mu}_k - \vec{\mu})^T \qquad (3.25)$$

où $\vec{\mu}_k$ est le vecteur moyen des éléments de la classe C_k .

Nous cherchons une transformation linéaire de matrice Ω qui maximise la dispersion des classes (mesurée par la matrice de dispersion interclasse) par rapport à la dispersion des éléments au sein des classes (mesurée par la matrice de dispersion intra classe). Une m re

de Fisher propose de choisir la transformation linéaire qui maximise le ratio :esure simple de dispersion est la valeur du déterminant des matrices de dispersion. Le critè

$$J_{LDA}(\Omega) = \frac{|S'_B|}{S'_W} = \frac{|\Omega^T S_B \Omega|}{|\Omega^T S_W \Omega|} \qquad (3.26)$$

où S'_B et S'_W sont les matrices de dispersion interclasse et intraclasse des projetés. On montre que les vecteurs colonnes de Ω sont les vecteurs propres ω_k qui correspondent aux plus grandes valeurs propres λ_k solutions de l'équation $S_B\omega_k = \lambda_k S_W\omega_k$. Si le déterminant de S_W vaut 1 (matrice isotropique), les vecteurs propres sont les vecteurs propres de S_B et sont dans la direction de $\vec{\mu}_k - \vec{\mu}$. La LDA revient à faire une ACP sur les vecteurs moyens des classes.

3.4 Techniques de réduction non- linéaires.

Les combinaisons linéaires sont particulièrement intéressantes car elles sont faciles à calculer. Cependant, il existe aussi des techniques non linéaires. Ce sont souvent des méthodes itératives qui permettent de trouver un nouvel espace de représentation, soit en utilisant des projections non linéaires, soit en recherchant un nouvel espace de représentation où les distances inter vecteurs sont sauvegardées. Les techniques les plus connues sont le *multidimensionnel scaling* (MDS), l'analyse en composantes curvilinéaires (ACC) **[Demartines et Hérault, 1997]** et la technique *FastMap* **[Faloutsos et Lin, 1995]**. Contrairement aux techniques linéaires, ces techniques permettent de mettre en évidence des classes qui ont des formes non-linéaires. Un exemple classique est une classe dont les points prennent la forme d'un fer à cheval. Pour ce type de classe, il n'est pas possible de réduire efficacement le nombre de dimensions par des techniques linéaires, car toutes les projections sur une droite perdent une grande partie de l'information.

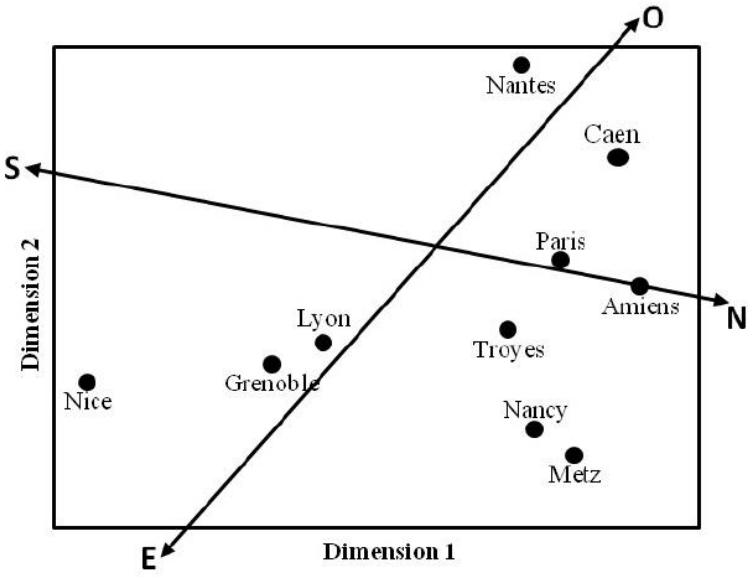
Une autre technique intéressante est décrite dans **[Lejsek et al, 2005].** Elle consiste à sélectionner les dimensions de manière aléatoire. Puis à déterminer si ces axes sont discriminants, sinon à reprojeter sur de nouveaux axes, jusqu'à ce que les axes deviennent discriminants. Cette technique est appliquée à la reconnaissance d'images par le contenu pour la détection de copie **[Lejsek *et al*, 2005]**.

[Jain et Zongker, 1997] propose une comparaison d'autres techniques de sélection des traits visuels. Voir également **[Vasconcelos, 2003, Goutte et al, 2006, Guyon et Elisseeff, 2003]**.

Les méthodes non linéaires ont pour objectif d'optimiser les représentations afin qu'elles reflètent au mieux la topologie initiale des données.

3.4.1 Isomap

Isomap **[Tenenbaum et al, 2000]** est une technique de réduction de dimensions qui, comme la méthode de positionnement multidimensionnel (MDS), part de la connaissance d'une matrice de dissimilarités entre les paires d'individus. Le but est cette fois de trouver une variété (non linéaire) contenant les données.



2003.

Figure 3. 5 - Positionnement de 10 villes françaises à partir de la matrice de leurs distances kilométriques

La figure 3.5. montre à gauche, un exemple des données non-linéaires et à droite le résultat de la représentation de ces points dans l'espace dirigé par les deux premières dimensions trouvées par l'application de l'algorithme isomap sur ces données.

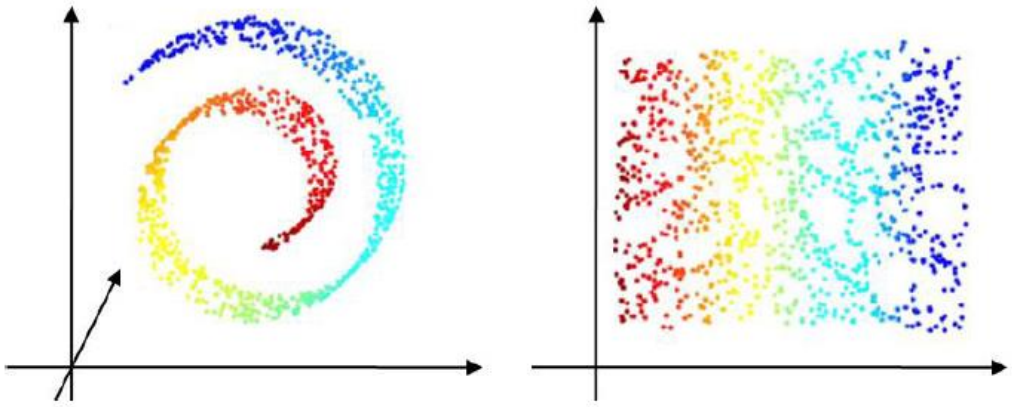


Figure 3. 6 : Isomap sur des données non-linéaires

Dans la suite, nous présentons une description des étapes de l'algorithme isomap. Supposons que nous ayons un ensemble de données $X = \{x_1, x_2, \dots, x_m\}$ composé de M observations où chaque observation $x_i = (x_{i1}, x_{i2}, .., x_{iN})$ est composée de N caractéristiques. Soit dune matrice symétrique de taille $M\ x\ M$ où chaque élément d_{ij} représente la distance entre x_i et x_i et $d_{ij} = 0$. L'algorithme Isomap, qui a X et d en entrée, passe en général par trois étapes:

1. Déterminer un « voisinage »pour chaque point x_i par la méthode des $k - plus$ proches calculés à partir des distances euclidiennes d en considérant que les distances euclidiennes approchent les distances géodésiques quet les vecteurs se trouvent à petite distance.
2. Estimer la matrice des distances géodésiques (D) entre tous les points. Isomap construit un graphe dont les sommets sont les points et les arêtes sont labelisées par les distances entre eux.
3. Appliquer la méthode MDS à la matrice de distances D pour obtenir un nouveau système de coordonnées euclidiennes qui préserve la géométrie intrinsèque de la variété.

3.4.2 Plongement localement linéaire

Plongement localement linéaire, LLE (Locally Linear Embedding) en anglais, est une méthode de réduction non-linéaire proposée en 2000 **[Roweis et Saul 2000].** Elle utilise une approche différente d'Isomap mais partage la même philosophie: les points dans un espace de haute dimensionnalité étant voisins doivent se retrouver proches dans une projection de faible dimension. Donc elle s'attache à ce que la disposition des plus proches voisins soit préservée.

LLE n'a pas besoin de calculer les distances entre tous les points de la variété. Il agit sur le voisinage de chaque vecteur pour découvrir la structure globale de l'ensemble. L'idée principale consiste donc à déterminer pour chaque vecteur x_i, point de l'espace contenant les données à projeter, ses k voisins calculés sur la base de la distance euclidienne et de considérer que dans ce voisinage, les données sont linéaires. Une fois les voisins de x_i déterminés, nous calculons les poids de reconstruction w_{ij} représentant le voisinage de chaque vecteur x_i associé à chaque couple (x_i, x_j), où x_j appartient au voisinage de x_i. Les w_{ij} permettront par la suite de reconstruire la topologie de voisinage de chaque vecteur x_i dans l'espace de projection.

Chapitre IV :

Contribution et Expérimentations.

4.1 Introduction

Au cours de la dernière décennie, avec les progrès des technologies informatiques et l'apparition du World Wide Web, il y'a eu une explosion de la quantité des données numériques générées, stockées, analysées et accessibles. Une grande partie de cette information est de nature multimédia, y compris le texte, l'image et des données vidéo. Les données multimédias sont généralement de très haute dimension, allant de plusieurs milliers à plusieurs centaines de mille.

La classification automatique de telles données est un domaine d'étude situé à l'intersection de deux thématiques de recherches majeures que sont l'analyse de données et l'apprentissage automatique (Machine Learning). L'optimisation de l'apprentissage automatique de telles bases de données est un objectif important car cela représente une grande part des problèmes de la classification automatique des données. Les limitations bien connues sont la mémoire et le temps de calcul. Les enjeux sont donc de trouver des algorithmes qui conduisent à des solutions les plus parcimonieuses possibles (ce qui a pour effet immédiat une réduction de la mémoire et du temps nécessaire pour la classification).

La plupart des techniques d'apprentissage automatique nécessitent de régler un nombre de paramètres pour l'apprentissage. Un nombre qui, pour la majorité des classificateurs (RN, Arbres de décision.. etc.), est en relation exponentielle avec la dimension de l'espace d'entrée. La formulation élégante des SVMs laisse très peu de place aux paramètres utilisateurs et le nombre de paramètres à régler est linéaire avec la taille de l'ensemble d'apprentissage. Dans le deuxième chapitre on a vu que l'apprentissage des SVMs se ramène à la maximisation d'une forme quadratique convexe sous des contraintes linéaires. Dans ces cas, il n'y a pas de problèmes de minimums locaux et la solution peut être trouvée en utilisant des algorithmes efficaces tels que la méthode d'optimisation par minimisation séquentielle **[Platt et al, 1999]** (Sequential Minimal Optimization :SMO) et l'algorithme de Joachims implémenté sous SVM_{light} **[Joachims, 1999]**.

Toutefois l'exécution de ces algorithmes demande un espace mémoire d'une taille lié linéairement à la taille de l'ensemble des exemples, ce qui augmente le temps d'exécution surtout lorsqu'on traite des bases de données de tailles importantes.

Nombreuses sont les heuristiques développées pour remédier cette faiblesse des SVMs, **[Panda et al,2006, Nguyen et Phung, 2010]**, parmi ces heuristiques celles qui effectuent une sélection de données d'apprentissage et dont l'objectif est d'atteindre un ensemble de données suffisants pour l'apprentissage sans parcourir toutes les données. Notre contribution fait partie de ces heuristiques.

Une première proposition dans ce travail développe une nouvelle procédure permettant d'appeler l'algorithme k-moyenne plusieurs fois pour réduire le nombre d'exemples nécessaires à l'apprentissage tout en préservant des résultats aussi proches que possible des résultats exacts.

D'autre part, les techniques de réduction de dimensionnalité ont été une solution clé dans de nombreux domaines de traitement de l'information, tels que l'extraction de données, la recherche d'information, et la reconnaissance des formes. Ainsi, pour accélérer plus l'apprentissage des SVMs nous avons proposé de réaliser la recherche du sous-ensemble suffisant pour l'apprentissage au sein d'un espace réduit du vecteur caractéristique généré par l'analyse linéaire discriminante (LDA), ce qui a comme effet d'améliorer et accélérer la recherche proposée par k-moyenne. Ci-après nous détaillons nos propositions et nous développons les expérimentations menées pour atteindre les objectifs de ces propositions.

4 .2 Mise en pratique des Séparateurs à Vaste Marge:

L'environnement de programmation choisi pour mettre en œuvre les différentes expérimentations réalisées dans ce chapitre est le Matlab. MATLAB est un logiciel de haut niveau et un environnement interactif pour les calculs numériques, la visualisation et la programmation.

4.2.1 Calcul des paramètres nécessaires au modèle SVM :

La réalisation d'un programme d'apprentissage par les SVMs se ramène à résoudre un problème d'optimisation impliquant un système de résolution dans un espace de dimension conséquente. L'utilisation de ces programmes revient surtout à sélectionner une bonne famille de fonctions noyau et à régler les paramètres de ces fonctions. Ces choix sont le plus souvent faits par une technique de validation croisée, dans laquelle on estime la performance du système en la mesurant sur des exemples n'ayant pas été utilisés en cours d'apprentissage. L'idée est de chercher les paramètres permettant d'obtenir la performance maximale. Si la mise en œuvre d'un algorithme des SVMs est en général peu coûteuse en temps, il faut cependant compter que la recherche des meilleurs paramètres peut requérir des phases de test assez longues.

4.2.2 Validation croisée

La validation croisée (Cross Validation en anglais) k-fold consiste à diviser l'ensemble de données en k partitions distinctes (dans nos expériences nous avons fixé k =5) de tailles presque égales pour en utiliser k−1 partitions pour l'apprentissage et la dernière pour évaluer la performance du classificateur. Ce processus est répété *k* fois pour chacun des *k* segments disponibles pour le test. L'erreur de test finale est la moyenne sur les tests croisés des k estimations possibles, elle est aussi appelée erreur de généralisation. A chaque test de validation, le paramètre du noyau σ et *C* changent suivant une grille de recherche prédéfinie.

4.2.3 Sélection des paramètres du modèle

La résolution de la méthode des machines à vecteurs support implique la sélection de plusieurs paramètres : le type de noyau, le ou les paramètres du noyau et le paramètre de régularisation C **[Vapnik, 1998, Vapnik, 2000].**

Dans le cas du noyau RBF utilisé dans nos expérimentations, deux paramètres sont à optimiser (c et σ), nous allons par exemple utiliser des séquences exponentielles:

$$C = 2^{-5}, 2^{-4}, \ldots, 2^{15} \text{ et pour } \sigma = 2^{-15}; 2^{-14} \ldots, 2^{3}.$$

Après une première évaluation, on peut relancer la procédure en raffinant la grille.

4 .3 Proposition d'une nouvelle approche de sélection de vecteurs supports:

Les SVMs sont l'une des techniques de noyau les plus récentes qui sont basées sur la programmation quadratique.

Rappelons la forme duale du problème quadratique à optimiser.

Trouver les multiplicateurs de Lagrange $\alpha_i \geq 0$ tels que:

$$\begin{cases} Max_{\alpha}\left\{\sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\,\alpha_j u_i u_j (x_i . x_j)\right\} \\ \alpha_i \geq 0, i = 1, \ldots, m \\ \sum_{i=1}^{m}\alpha_i u_i = 0 \end{cases}$$

On remarque que :

- En pratique, seuls les points qui sont sur les hyperplans frontières jouent un rôle, car les multiplicateurs de Lagrange sont non nuls pour ces points seulement.
- Ils sont appelés *vecteurs de support* ou *exemples critiques*.
- Le vecteur solution w^* a donc une expression en termes d'un sous-ensemble des exemples d'apprentissage : les exemples critiques.
- C'est en même temps intuitivement satisfaisant puisque l'on voit bien que l'hyperplan solution est entièrement déterminé par ces exemples.
- La solution ne dépend plus de la dimension *d* de l'espace d'entrée, mais de la taille *m* de l'échantillon de données et même du nombre *mc* d'exemples critiques qui est généralement bien inférieur à *m*.
- Les méthodes d'optimisation quadratique standards suffisent pour la plupart des cas.

Si nous supposons un ensemble d'entraînement de taille N alors la complexité temporelle est $O(n^3)$ et la complexité spatiale est en $O(n^2)$. Lorsqu'on applique les SVMs multi-

classe, la complexité est en mesure de dépasser le double. Ce problème est particulièrement important pour les grandes bases de données. Dans ce cas en général un grande nombre de vecteurs supports (SV) est choisi, ce qui augmente le temps nécessaire pour calculer la matrice noyau et par conséquent la réponse du système. Ce problème peut être surmonté par la réduction du nombre de SVs.

Notre approche de sélection de données consiste à utiliser le regroupement k-moyenne plusieurs fois au sein d'une procédure itérative permettant à son issue d'obtenir un ensemble de vecteurs proches des frontières de décision. Cet ensemble est considéré comme l'ensemble des vecteurs supports suffisants pour exécuter le SMO .L'algorithme K-moyenne est utilisé plusieurs fois avec différentes valeurs du taux de compression (nombre de centres différents.).

L'idée est de commencer par traiter des clusters de grandes tailles (d'un taux de compression faible) tout en augmentant le taux de compression d'une itération à une suivante, la conséquence en est que la tailles des clusters diminue à chaque fois qu'on se rapproche plus des frontières de décision. Donc à chaque itération on exécute le k-moyenne avec un nouveau taux de compression tout en remplaçant les données des clusters homogènes par leurs centres et en gardant les données des clusters mixtes. L'ensemble de données obtenues est alors considéré comme le nouvel ensemble d'apprentissage à utiliser à la prochaine itération.

- Le taux de compression (TC) est calculé par la formule suivante :

Taux de Compression = Taille de données / Nombre de centres

- **L'algorithme suivant résume les étapes de la procédure :**

Entrées:

- L'ensemble d'apprentissage initial (T).
- La valeur taux de compression initial (TCI).
- La valeur du taux de compression final (TCF).
- La valeur du pas du taux de compression (PTC).

Sorties:

- L'ensemble d'apprentissage réduit à utiliser par le SMO (T).

Etape 1:

- Définir le taux de compression initial qui est à l'entour de 30% de taille de d'apprentissage.
- TC (taux de compression) = TCI (taux de compression Initiale) .

Etape 2:

- Appliquer le k-moyennes à l'ensemble **T** avec le taux de compression **TC**.

Etape3 : Partager les clusters obtenus en deux groupes :

- Les Clusters Homogène dont les données ont le même label de classe notés H et l'ensemble de leurs centres noté : $\mathbf{C^H}$.
- Les Clusters Mixtes dont les données n'ont pas le même label de classe notés M et l'ensemble de leurs centres est noté : $\mathbf{C^M}$.

Etape4 : De-clustring : récupérer les données appartenant aux clusters mixtes.

Etape5 : Mise à jour de l'ensemble d'apprentissage T par l'union de l'ensemble de données des clusters mixtes et l'ensemble des centres des clusters Homogènes :

$\mathbf{T}= \{ \mathbf{C^H} \cup \mathbf{M} \}$.

Etape6 : Augmenter le taux de compression TC = TC +PTC.

Etape7 : Si TC > TCF alors terminer ; sinon aller à l'étape 2.

Etape8 : Appliquer le SMO au dernier ensemble d'apprentissage obtenue T.

4.4 Résultats expérimentaux de l'approche de sélection par K-moyenne:

Afin de valider notre approche, nous présentons maintenant nos expérimentations réalisées en utilisant des bases de données du répertoire UCI : http://www.ics.uci.edu/.
Dans une première étape, nous nous sommes limités aux problèmes à deux classes qui ont donné des résultats de référence. Deux bases de données ont été alors utilisées; la première est Ripley et la deuxième est Banana.

4.4.1 Résultats Expérimentation de la base Ripley :

Cette base de données est destinée aux tests des classificateurs binaires. Elle est composée de 250 exemples dont les vecteurs caractéristiques sont de taille 2.
Initialement pour prouver l'importance de la récupération des zones mixtes nous avons exécuté une itération de l'approche proposée en utilisant la base Ripley. Donc après une exécution de l'algorithme K-moyenne nous avons récupéré toutes les données des clusters Mixtes et que les centres des clusters Homogènes. L'union de ces deux ensembles constitue l'ensemble d'apprentissage du SMO :
Les paramètres de la fonction Kernel σ et *C* paramètre ont été fixés par validation croisée à σ =1 et *C* = 10 (cf. section 4.2 de ce chapitre), les résultats sont résumés dans le tableau 4.1 :

	SVMs	**K-moyenne + SVMs**				
		K=10	**K=20**	**K=30**	**K=40**	**K=50**
Temps CPU (s)	0.4138	0.1800	0.1743	**0.1519**	0.1883	0.1909
Erreur	**0.0930**	0.1150	0.1000	**0.0950**	0.1050	0.0980

Tableau4. 1. : Résultats obtenus de l'exécution d'une itération de la procédure de sélection proposée.

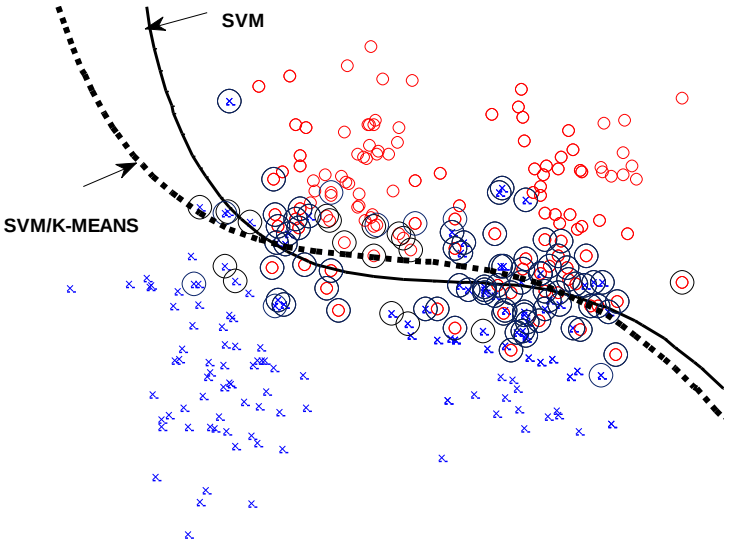


Figure 4. 1. : Frontière de décision d'une seule itération pour k =30 .

La deuxième expérimentation consiste à appliquer l'approche de sélection proposée avec les valeurs suivantes des paramètres nécessaire à son exécution : TCI = 0 .1, TCF =0.7 et PTC = 0.05. Les résultats obtenus à chaque itération sont résumés dans le tableau 4.2. Avec :

- **TC** est le taux de compression de chaque itération,
- **K** est le nombre des clusters générés par le K-moyenne.
- **TrnSize** est la taille de l'ensemble d'apprentissage réduit à chaque itération.

Iteration	**1**	**2**	**3**	**4**	**5**	**6**
CR	0.1	0.15	0.20	0.25	0.30	0.35
K	25	20	29	38	39	44
TrnSize	250	152	143	140	131	131
Iteration	**7**	**8**	**9**	**10**	**11**	**12**
CR	0.40	0.45	0.50	0.55	0.60	0.65
K	45	58	62	73	78	91
TrnSize	129	127	124	124	118	112

Tableau 4. 2. : Résultats détaillés de l'approche de sélection proposée**.**

Après la dernière itération nous avons appliqué le SMO au dernier ensemble d'apprentissage réduit à 112 exemples. La comparaison des résultats de l'exécution du SMO sans et avec réduction est représenté dans le tableau 4.3. .Quatre paramètres sont calculés :

- **TRNsize** : La taille de l'ensemble d'apprentissage,
- **NSV** : Le nombre des vecteurs supports.
- **CPU Time (s):** Le temps d'exécution.
- **Error**: Le taux d'erreur.

	SVMs	**SVMs/K-EMANS**
TRNsize	250	112
NSV	116	45
CPU Time (s)	0.4138	0.1588
Error	0.0930	0.0940

Tableau 4. 3. : Résultats obtenus pour Ripley.

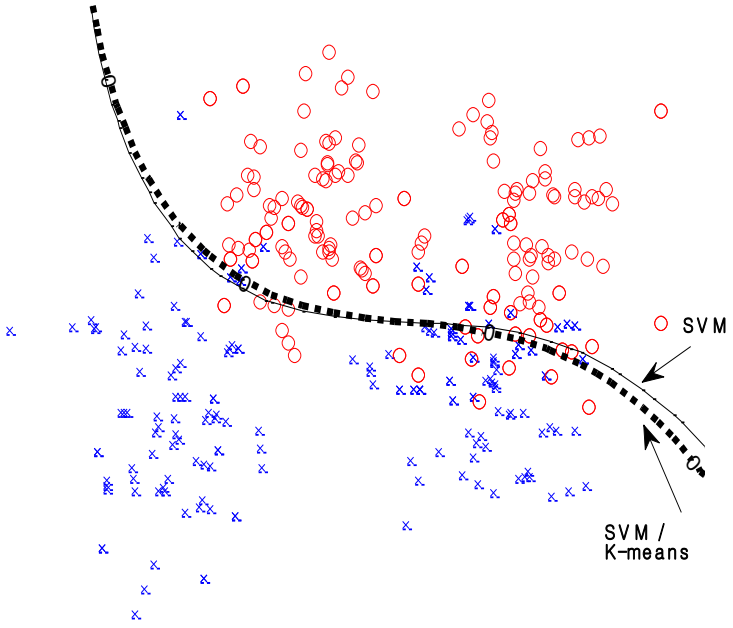


Figure 4. 2. : Comparaison des frontières de décision.

- **Conclusion 1**

Les résultats obtenus en utilisant la base Ripley sont importants **[Chaibi, 2013]**.
Premièrement, le tableau 4.1. nous a permis d'évaluer l'efficacité de la sélection par k-moyenne. Dans le tableau 4.2. nous avons testé notre proposition en augmentant à chaque fois le taux de compression d'un pas de 0.05. Ce paramètre influe directement sur la taille des données à supprimer à chaque itération. Nous avons aussi remarqué qu'en augmentant la valeur du taux de compression le nombre des clusters augmente ce qui signifie que leurs tailles diminuent sûrement d'une itération à une autre. Ce choix ciblé garanti le fait de traiter soigneusement les données tout en prenant compte de leur proximité des frontières de décision.
Il est alors clair que les objectifs de la proposition ont été atteints : la taille de l'ensemble d'apprentissage à diminué de 250 à 112, le nombre des vecteurs supports a diminué de 116 à 45 et le facteur temps d'exécution a beaucoup diminué (cf. Tableau 4.3.). En contrepartie nous n'avons pas beaucoup perdu en précision et les frontières de décision obtenues (cf. Figure 4.2.) coïncident presque quasiment avec les frontière des SVMs.

Toutefois, il a été constaté que la petite taille de la base ne permet pas de critiquer le fond de l'approche et c'est pour cette raison qu'il s'est révélé nécessaire de recourir à d'autres bases de tailles plus grandes.

4.4.2 Résultats expérimentaux de la base Banana :

La deuxième base de données utilisée est Banana composée de 4770 exemples d'apprentissage et 530 exemples de test. La taille du vecteur caractéristique est 2. Les paramètres nécessaires au déroulement de la procédure sont CRI = 0 .1, CRF =0.7 et ICR = 0. 1
Les résultats des itérations de la procédure proposée sont résumés dans le tableau 4.4. avec :

- **TC** : Le taux de compression,
- **K** : Le nombre des centres générés par le k-moyenne.
- **TrnSize** : La taille de l'ensemble des données réduits pour l'apprentissage SMO.

Iteration	**1**	**2**	**3**	**4**	**5**	**6**	**7**
CR	0.01	0.11	0.21	0.31	0.41	0.51	0.61
K	48	408	378	427	449	441	421
TrnSize	4770	3708	1797	1377	1095	864	789

Tableau 4. 4. : Résultats des itérations de la procédure de sélection pour Banana.

Dans les trois figures suivantes il est illustré les résultats détaillés de la troisième itération de la procédure exécutée à Banana. La figure 4.4. représente l'ensemble d'apprentissage obtenu après la deuxième itération, ensuite dans la figure 4.5. les points en gras sont les éléments du nouvel ensemble sélectionné pour la prochaine itération. La figure 4.6. illustre l'ensemble d'apprentissage issu de la troisième exécution en éliminant les zones Homogènes, cet ensemble de données est le nouvel ensemble à traiter en quatrième itération.

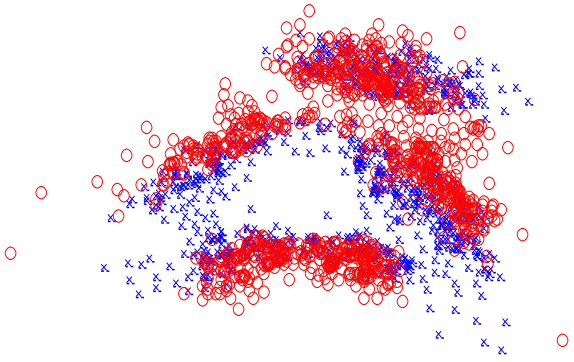


Figure 4. 3. : L'ensemble d'apprentissage de la troisième itération

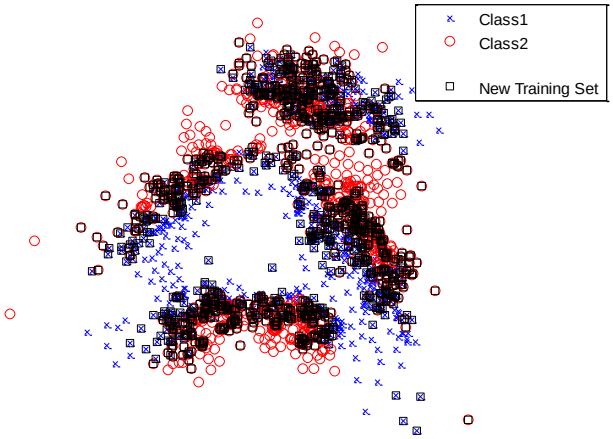


Figure 4. 4. : L'ensemble de données à supprimer en gras.

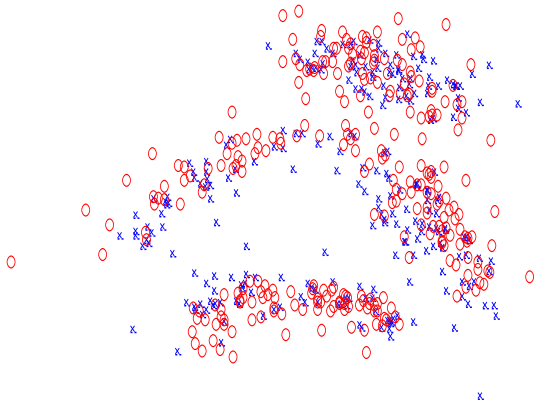


Figure 4. 5. : L'ensemble d'apprentissage à traiter en quatrième itération.

Après la dernière itération nous avons appliqué le SMO au dernier ensemble d'apprentissage réduit, composé de 789 exemples qui représente 17% de l'ensemble d'apprentissage.

	SVM	SVM/K-MEANS
TRN size	4770	789
NSV	**1107**	**742**
CPU Time (s)	359.2292	30.5782
Error	**0.1019**	**0.1113**

Tableau 4. 5. : Comparaison des Résultats obtenus.

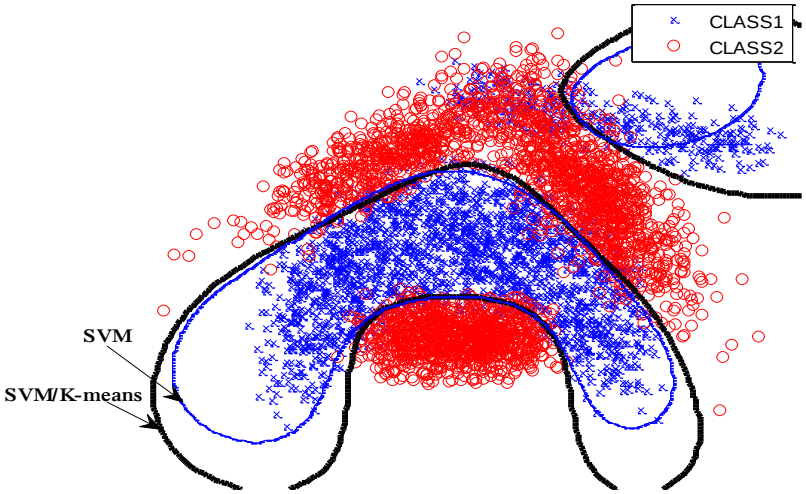


Figure 4. 6. : Comparaison des frontières de décisions .

- **Conclusion 2**

Les résultats obtenus pour Banana, résumés dans le tableau 4.5., démontrent de nouveau l'efficacité de la procédure de sélection proposée. L'ensemble d'apprentissage se trouve réduit de 4770 à 742, ce qui a entrainé simultanément un gain considérable du temps d'exécution estimé à 90%. D'autre part la frontière de décision obtenue dans la figure 4.6. renforce encore cette déduction.

4.4.3 Autres bases de données:

Pour tester davantage notre approche, il s'est révélé important d'augmenter la complexité des bases de données utilisées et à cet effet il est nécessaire de recourir à d'autres bases du répertoire UCI http://www.ics.uci.edu/. . Le tableau 4.6. donne certains détails sur la description de ces ensembles de données.

Datasets	**Feature vector size**	**N° Classes**	**Training set size**	**Testing set size**
Led	24	26	1000	1000
Credit	20	6	600	400
VEHICULE	18	4	700	147
Abalone	8	29	3177	1000
Deterwing	10	11	528	462
Wav	21	3	750	250

Tableau 4. 6. : Caractéristiques des bases de données UCI choisies.

Les résultats obtenus sont montrés dans le tableau 4.7. :

	SVM				**SVM/KMEANS**			
	TRN	**Nsv**	**Error**	**CPU Time**	**TRN**	**Nsv**	**Error**	**CPU Time (s)**
Deterding10	528	376	0.09848	15.541	69	69	0.1122	7.4976
Wav 21	750	500	0.07640	25.940	68	68	0.1245	13.1388
Vehicule18	700	537	0.06603	21.446	128	128	0.1174	11.9656
Credit20	600	600	0.11120	20.233	242	239	0.1408	10.1700
LED24	1000	1000	0.0990	37.820	353	352	0.1586	15.7.16
Abalone8	3177	3074	0.1095	40.225	1285	1285	0.1244	18.142

Tableau 4.7. : Résultats obtenus.

- **Conclusion 3 :**

Les expérimentations réalisées sur les différentes bases de données utilisées, ont permis réduction efficace de l'ensemble d'apprentissage. Toutefois elles ont dévoilé les insuffisances de l'approche en augmentant la taille du vecteur caractéristique. Ceci est lié particulièrement à la complexité de l'algorithme du k-moyenne qui est en relation linéaire avec la taille du vecteur caractéristique et le nombre de classes. Pour remédier à ce problème une extension de la proposition est détaillée dans la prochaine section.

4.5 Sélection de données au sein d'un espace LDA :

La complexité temporelle de l'algorithme K-moyenne est calculé par la formule (D*N*K*I) avec D la taille du vecteur caractéristique, N le nombre d'échantillons est le nombre de classes et le nombre des itérations.

Les approches par partitionnement, dont l'algorithme des k-moyennes **[MacQueen, 1967]** en est l'un des plus importants représentants. Il consiste le plus souvent à construire une collection de classes disjointes formant une partition des données par optimisation d'un critère objectif qui est dans notre approche la distance euclidienne. Plus la dimensionnalité augmente, moins la distance euclidienne a de sens. En effet, en dimension *d*, la distance entre les exemples *x* et *y* est :

$$\|x, y\| = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

Par conséquent, plus *d* est grande, plus les dimensions dans lesquelles les deux exemples sont proches deviennent négligeables par rapport au total de celles dans lesquelles ils sont loin. Ce sont seulement les très petites distances qui sont informatives. En haute dimension : on peut alors dire que les exemples sont proches dans toutes les dimensions. Ainsi, les méthodes basées sur la distance entre les exemples comme le *k-moyenne* fonctionnent en faible dimension, mais elles verront leurs performances se dégrader rapidement en haute dimension.

Il est alors clair qu'utiliser l'algorithme K-moyenne pour accélérer la réponse des SVMs semble insuffisant pour les bases de données de tailles importantes. Dans cette section nous proposons une amélioration de notre sélection en l'appliquant dans un espace réduit de données obtenu par projection des données dans un espace. L'effet important derrière cette proposition est que la projection choisie permet en premier lieu d'améliorer la qualité de la recherche proposée et en deuxième lieu d'accélérer cette recherche tout en préservant un maximum d'information utile.

La LDA effectue une véritable séparation des classes en minimisant les variations entre les exemples d'une même classe tout en maximisant les variations entre les exemples de classes différentes. Cette méthode de réduction de données réalise une projection de données dans le sens du vecteur qui maximise la séparabilité des classes .Donc dans le nouveau espace, les exemples d'une même classe sont les plus proches possibles les uns des autres et les exemples des classes différentes sont les plus éloignés possibles les uns des autres.

4.5.1 Calcul du modèle LDA :

Pour calculer le modèle LDA associé à chaque base nous considérons que les données suivent la loi Gaussienne qui est bien adaptée à beaucoup de phénomènes physiques et naturels. Elle est simplement décrite par sa moyenne μ_k et sa matrice de covariance Σ est donnée par :

$$f_{ik}\,(x_i;\mu_k,\Sigma)6 = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}}\ \exp(-\frac{1}{2}(x_i-\mu_k)^T\ \ \Sigma^{-1}(x_i-\mu_k)\ \ )\quad (4.1)$$

<u>Apprentissage du modèle LDA</u>

L'apprentissage du modèle LDA consiste en l'estimation des paramètres μ_k et Σ à partir des données d'apprentissage. Ces estimations sont données par :

$$\pi_k = \frac{\Sigma_{i/z_i=k}}{n} = \frac{n_k}{n} \qquad (4.2)$$

où n_k est le cardinal de la classe *k*, z_i est l'étiquette de classe de l'exemple x_i et z_i = k désigne le fait que l'exemple x_i appartient à la classe k. L'estimation des paramètres μ_k

(les moyennes) et Σ (matrice de covariance) à partir des données d'apprentissage sont données par :

$$\mu_k = \frac{1}{n_k}\sum_{i|z_i=k} x_i \quad (4.3)$$

Et

$$\Sigma = \frac{1}{n_k}\sum_{k=1}^{K}\sum_{i|z_i=k}(x_i - \mu_k)(x_i - \mu_k)^T \quad (4.4)$$

Algorithme de l'analyse Discriminante Linéaire (LDA) :

Input:

- Données d'apprentissage;X^{train} = (x_1^{train}, x_2^{train}..., x_n^{train});
- classes des données d'apprentissage z^{train} = (z_1^{train}, z_2^{train}... , z_n^{train})

Algorithme **train_LDA** :

for k ← 1 to K do

| Calculer la proportion π_k de la classe k en utilisant l'équation (4.2)

| Calculer la moyenne μ_k en utilisant l'équation (4.3)

end

Calculer la matrice de covariance des classes Σ en utilisant l'équation (4.4)

Result: Paramètres (π_1,... ,π_K,μ_1,... ,μ_K, Σ)

4.5.1 Résultats expérimentaux de la combinaison LDA/K-moyenne pour la sélection des vecteurs supports :

Dans un premier temps, nous allons tester notre dernière stratégie sur quatre bases de données choisies : Svmguide1, Svmguide3, Wave et Letter.

Svmguide1 est composée de 3084 exemples pour l'apprentissage et 4000 exemples pour le test. La taille du vecteur caractéristique est 4. Les valeurs des paramètres obtenus pour cette base sont C = 5, et σ = 0.125 et dans l'espace réduit : C =24, et σ = 0.025.

	La classification SVM			
	Ensemble d'apprentissage	**Selection par K-moyenne**	**L'èspace LDA**	**Selection par k-moyenne dans l'espace LDA**
TrnSize	3084	325	3084	309
Nsv	**330**	228	532	**184**
CpuTime	**94.3962**	15.8054	20.2469	**7.4569**
Error	**0.0325**	0.0525	0.0327	**0.0362**

Tableau 4. 8. : Résultats obtenus pour Svmguide1.

Svmguide3 composée de 1242 exemples d'apprentissage et 41 exemples de test. La taille du vecteur caractéristique est 21. Le nombre de classes est deux. Les valeurs des paramètres obtenus pour cette base sont C = 128, et σ =0.5 et dans l'espace réduit ces paramètres ont été fixés à C = 16 et σ = 0.001.

	La classification SVM			
	Tout l'ensemble d'apprentissage	**Sélection par K-moyenne**	**L'espace LDA**	**Sélection par k-moyenne dans l'espace LDA**
TrnSize	1242	771	1242	581
Nsv	**936**	641	1184	**569**
CpuTime(s)	**153.0779**	42.1227	94.3541	**11.7407**
Error	**0.008**	0.0732	0.0201	**0.0244**

Tableau 4. 9. : Résultats obtenus pour Svmguide3**.**

Wave composée de 750 exemples d'apprentissage et 250exemples de test. Chaque vecteur caractéristique est composé de 21 instances. Le nombre de classes est 3. Les paramètres du classificateur SVM ont été fixés à C = 256, et σ = 1, dans l'espace réduit à C = 0.5, et σ = 0.5.

	La classification SVM			
	Tout l'ensemble d'apprentissage	**Sélection par K-moyenne**	**L'espace LDA**	**Sélection par k-moyenne dans l'espace LDA**
TrnSize	750	199	750	199
Nsv	**83**	49	80	**25**
CpuTime (s)	**1.5600**	1.4653	0.2338	**0.1824**
Error	**0.1360**	0.1920	0.1400	**0.1400**

Tableau 4. 10. : Résultats obtenus pour Wave**.**

Letter composée de 15000 exemples d'apprentissage et 5000 exemples de test. Le vecteur caractéristique est de taille 16; le nombre de classes est 26. Les paramètres du noyau SVM ont été fixé à. C =10 et σ =0.125, dans l'espace réduit à C =64 et σ =2.

	La classification SVM			
	Tout l'ensemble d'apprentissage	**Sélection par K-moyenne**	**L'espace LDA**	**Sélection par k-moyenne dans l'espace LDA**
TrnSize	15000	9039	15000	5767
Nsv	**10467**	7627	12281	**5592**
CpuTime (s)	**1165.4**	640.257	966.9900	**150.6395**
Error	**0.0816**	0.1200	0.0912	**0.1034**

Tableau 4. 11. : Résultats obtenus pour Letter**.**

- **Conclusion 4 :**

Les résultats obtenus pour les différentes bases de données ont permis de conclure que la réalisation de la recherche au sein d'un espace généré par LDA a été rentable, notamment dans la réduction de la taille de l'ensemble des exemples critiques et le gain de temps de traitement. Cependant la sélection par k-moyenne se trouve légèrement affectée **[Chaibi et Laskri, 2015]**.

4.6 Application à la reconnaissance des Mots Arabes manuscrits :

L'objectif de cette dernière expérimentation est de confronter le problème de la reconnaissance des mots arabes manuscrits hors-ligne en utilisant les Machines à Vecteurs Support ou SVMs (Support Vector Machines). En général, la plupart des méthodes d'apprentissage de machine comme Réseau de Neurones (RN) et les Chaines de Markov Cachées(CMC) possèdent un grand nombre de paramètres d'apprentissage à fixer par l'utilisateur. En contrepartie, la formulation élégante de SVM laisse très peu de place aux paramètres utilisateurs.

Notre projet s'inscrit donc dans le cadre de reconnaissance de mots arabes non contraintes **[Ayat et al, 2001]** par les séparateurs à vaste marge. Il s'agit de classifier chaque mot parmi un ensemble de mots reconnaissables par le classificateur SVM. La phase d'apprentissage de ce classiffieur constitue la phase fondamentale dans tout le processus de la reconnaissance. Le schéma de la figure 4.8. résume les étapes nécessaires pour la réalisation d'un système de reconnaissance des mots par SVM.

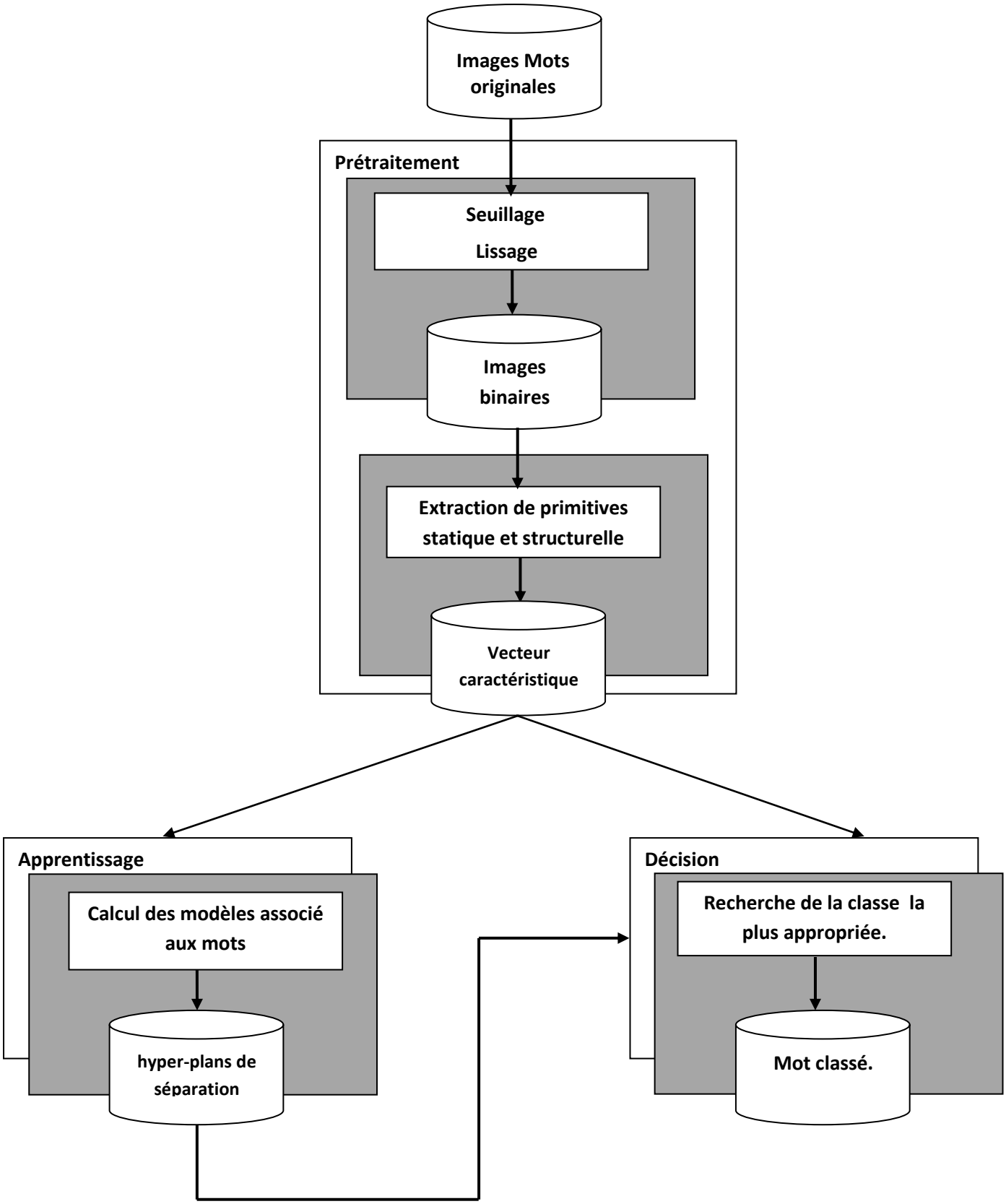


Figure 4. 7. : Schéma général des étapes importantes de la classification des mots arabes par SVM.

4.6.1 Bases de données des noms des Wilayas Algériennes:

Pour tester notre proposition, un vocabulaire restreint a été choisi correspond aux 48 noms des wilayas algériennes. Cette base de données conçue au sein du **LAB**oratoire de recherche sur la **G**estion **E**lectronique de **D**ocuments LABGED de l'Université de Annaba a été utilisée dans plusieurs travaux importants **[Souici et Sellami, 2002, Farah et al, 2005, Azizi et al, 2010]**. Cette base est extraite de 300 images écrites par 100 scripteurs où chaque scripteur reprend l'écriture de chaque nom trois fois. Les feuilles sont scannées ensuite divisées en plusieurs images chacune contient un mot manuscrit et elle est d'une taille 288*86 pixels.

4.6.2 Prétraitement de l'image :

Après l'acquisition des images par un scanner sous le format .bmp, un ensemble de rétraitements y sont appliqués pour la préparer à la phase d'analyse qui comporte :

- La binarisation (seuillage) : consiste à transformer l'image en niveau de gris en une image bimodale (noir et blanc). Cette image binaire est trouvée en utilisant un seuil déduit de l'histogramme de l'image en niveaux de gris.
- Le Lissage : en utilisant un masque 3* 3 nous pouvons éliminer les bruits dus aux conditions d'acquisitions par Bouchage et Nettoyage.

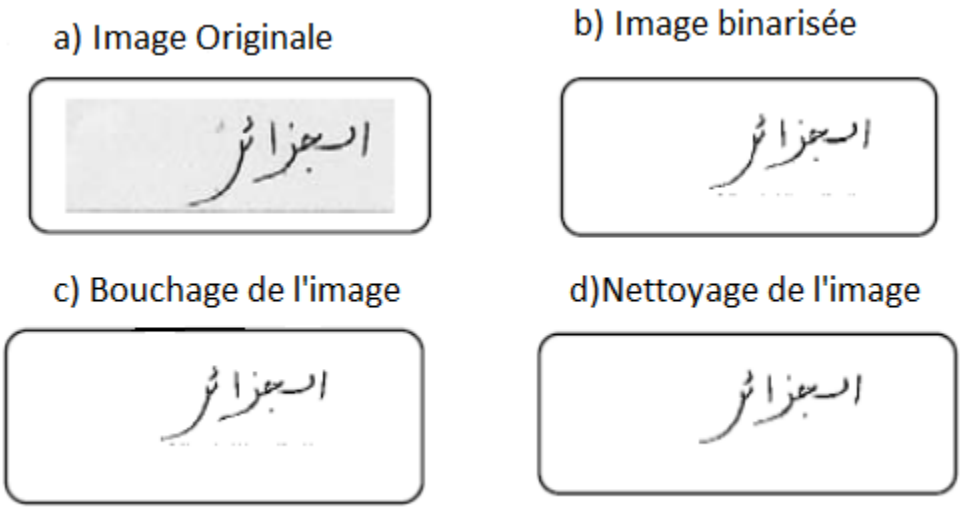


Figure 4. 8. : Données en préservant uniquement les caractéristiques essentielles de la forme.

- Détection de la ligne de base : pour extraire facilement les caractéristiques structurelles. La ligne de base définit comme étant la ligne la plus dense en point

noir et correspond à la plus grande valeur de l'histogramme horizontale de l'image.

4.6.3 Les Caractéristiques Structurelles :

Les caractéristiques structurelles globales retenues dans notre étude sont au nombre de 9:

- Le nombre de composantes connexes : groupes de pixels noirs connectés.
- Le nombre de boucles (occlusions) : est égal au nombre de fonds blancs détectés dans l'image moins un.
- Le nombre d'ascendants (hampes) et de descendants (jambages) : représenté par une montée ou une descente qui se termine en dehors de la zone médiane déterminée à partir de la ligne de base.

Les points diacritiques : détectés à partir de la définition de l'air d'un point diacritique sont :

- Le nombre de point en haut;
- Le nombre de points en bas ;
- Le nombre de deux points liés en haut;
- Le nombre de deux points liés en bas ;
- Le nombre de trois points liés en bas ;

4.6.4 Les caractéristiques statistiques :

Au cours des cinq dernières années l'intérêt s'est porté sur les moments orthogonaux tels que les moments de Legendre ou les moments de Zernike. Les moments de Zernike possèdent une propriété d'orthogonalité permettant la reconstruction de la forme. Les polynômes de Zernike furent proposés en 1934 par Zernike **[Zernike, 1934]**. Dérivés de ces polynômes, les moments ont été utilisés par de nombreux auteurs en reconnaissance d'écriture.

4.6.7 Résultats expérimentaux de la sélection proposée :

La phase d'apprentissage consiste à calculer les hyperplans de séparation associés aux différentes classes des mots de la base. Le nombre des hyperplans à désigner est donc égal au nombre de wilayas de l'Algérie qui est 48. L'ensemble d'apprentissage est d'une

taille égale à 300*48 =14400 exemples. Les résultats obtenus pour la reconnaissance des noms des Wilayas algériennes sont résumés dans le tableau4.12

	SVM	**SVM/K-MEANS**	**SVM/Kmeans/LDA**
TRN size	14400	6906	4530
NSV	**10503**	**6032**	**4186**
CPU Time (s)	359.2292	207.5782	80.5782
Error	**0.096**	**0.1313**	**0.102**

Tableau 4. 12. : Comparaison des Résultats obtenus**.**

Conclusion

Ce chapitre est consacré aux détails de notre proposition d'accélération de l'apprentissage des SVM laquelle s'est articulée autour de deux aspects importants, le premier consiste en la réalisation d'une réduction de la taille de l'ensemble d'apprentissage par sélection de données en utilisant le regroupement K-moyenne, et le deuxième est l'accélération de cette sélection en l'appliquant dans un espace réduit du vecteur caractéristique généré par l'analyse discriminante linéaire de Fisher.

Les résultats obtenus sont originaux et révèlent l'importance de la proposition de sélection de vecteurs supports au sein d'un espace réduit pour augmenter l'efficacité du classificateur SVM.

Conclusion et perspectives

Pendant ces dernières années, un intérêt remarquable a été accordé aux Séparateurs Vaste Marge (SVMs) **[Vapnik, 1995]**. Ces algorithmes d'apprentissage, grâce à leur pouvoir de généralisation robuste, ont trouvé des domaines d'application très variés comme la reconnaissance de caractères manuscrits **[Gherghout et Souci, 2009**] et la détection de visages **[Heisele et al, 2001].** Dans cette thèse, une nouvelle approche utilisant les SVMs pour la reconnaissance des mots arabes manuscrits a été développée.

La difficulté de la reconnaissance de l'écriture arabe a amené plusieurs chercheurs à conduire plusieurs travaux pour mettre en œuvre une solution adéquate. Parmi ces travaux marquants ceux réaliser au sein du laboratoire LABGED[9] de l'Université Badji Mokhtar Annaba, en utilisant les Réseaux de Neurones (RN) **[Souci et Sellami, 2002,Farah et al, 2005, Azizi et al, 2010].** Toutefois, la mise en œuvre de ces classificateurs neuronaux exige l'ajustement de plusieurs paramètres, ce qui n'est pas le cas des SVMs. Cependant, l'utilisation des SVMs était limitée à un groupe de chercheurs car les algorithmes d'apprentissage étaient longs et difficiles à implémenter **[Chaibi, 2013]**. Pour palier à ce problème, des approches analytiques ont été développées tels que l'optimisation par minimisation séquentielle **[Platt et al, 1999]** (Sequential Minimal Optimization :SMO) et l'algorithme de Joachims implémenté sous SVM_{light} **[Joachims, 1999]**. Mais la mise en œuvre de ces approches pour le traitement des problèmes de grandes tailles s'est révélée coûteuse en espace mémoire et en temps d'exécution, et les outils existants ne permettent pas de dépasser cette carence.

Donc, le travail réalisé dans cette thèse se situe dans le cadre de l'optimisation dans le sens d'accélération de l'apprentissage des Machines à Vaste Marge pour le traitement des bases de données de grandes tailles et plus particulièrement pour le traitement des bases de reconnaissance de mots arabes manuscrits. Notre domaine d'intérêt de recherche touche aussi d'autres disciplines telles que les systèmes à temps réel et la détection d'intrusion…etc.

9 : LABGED : LABoratoire de recherche sur la Gestion Electronique de Documents -UBMA-

Notre approche consiste à atteindre une phase de sélection des vecteurs supports avant l'exécution des approches analytiques; cette sélection appelle l'algorithme k-moyenne pour différencier entre les zones contenant les vecteurs supports (représentées par les clusters mixtes) et les zones ne contenant pas de vecteurs supports (représentées par les clusters homogènes) ces dernières zones sont les zones à supprimer.

Les zones mixtes issues de la dernière itération de la procédure représentent alors l'ensemble contenant les vecteurs supports suffisants pour obtenir des frontières aussi proches que possible des frontières exactes, ce qui se traduit par une perte négligeable en précision. Il est à souligner que notre proposition dépend fortement du choix des valeurs des paramètres de la procédure de sélection proposée car ils influent directement sur la taille et la qualité des données à supprimer à chaque itération.

La sélection proposée prend plus de flexibilité lorsqu'elle est réalisée dans un espace réduit des données obtenu par projection dans le sens d'une direction permettant une meilleure séparabilité de classes. Pour ce faire nous avons opté pour la projection dans un espace généré par l'Analyse Discriminante Linéaire (LDA).

Nous avons implémenté l'approche proposée en langage Matlab. Les résultats obtenus sont importants pour différentes bases de données.

Pour évaluer la performance de notre proposition en termes de rapidité d'apprentissage et de précision de reconnaissance nous avons mené une étude expérimentale qui porte sur des mots manuscrits numérisés arabes. Les résultats obtenus montrent que notre système a permis de reconnaître une bonne partie de l'échantillon du test tout en évitant l'apprentissage lourd des SVMs .

Le travail réalisé nous ouvre plusieurs perspectives parmi lesquelles nous citons :

- L'étude de la possibilité d'améliorer la performance de notre proposition par l'utilisation d'acronymes du k-moyenne tel que k-méloïdé.

- Le remplacement de notre condition d'arrêt de la procédure de sélection par un autre critère d'arrêt pour éviter le plus possible l'élimination des exemples importants à la précision du classifier ou bien le contraire.

- La réalisation de notre projection de l'espace de donnée dans d'autres espaces et l'évaluation de l'effet de cette projection sur la précision du système.

- Certains aspects liés directement aux systèmes de reconnaissance de l'écriture n'ont pas été abordés et doivent être examinés ultérieurement dans le cadre du suivi de cette thèse. C'est le cas des différentes phases de l'acquisition d'image, de pré-traitement, des techniques de segmentation, de définition et d'extraction de caractéristiques pour la classification et des méthodes de post-traitement qui permettent souvent d'améliorer les résultats de reconnaissance.

- Il serait aussi intéressant d'étendre notre approche de reconnaissance de mots vers la reconnaissance de phrases et de textes.

Références Bibliographiques

[Aizerman et al,1964] A. Aizerman,, E.M. Braverman, L. Rozoner, Theoretical foundations of the potential function method in pattern recognition learning. Autom. Remote Control. 1964, 25, 821–837.

[Ayat, et al, 2001] N.E. Ayat, M. Cheriet, L. Remaki, et c.Y. Suen. Kmod, " New Support Vector Machine Kernel for Pattern Recognition. Application to digit image recognition". In Proceedings of the IEEE International Conference on Document Analysis and Recognition, pages 1215-1219, Seattle,USA, Sept. 2001.

[Azizi et al, 2002] N. Azizi, T. Sari, L. Souici-Meslati, et M. Sellami, "Une architecture de combinaison floue de classifieurs neuronaux pour la reconnaissance de mots arabes manuscrits", Proceedings du 3ème Colloque International Francophone sur l'Ecrit et le Document (CIFED), Hammamet-Tunisie, pp. 89-96, 2002.

[Azizi et al, 2010] N. Azizi, N. Farah, M. Sellami, et A. Ennaji, "Using diversity in classifier set selection for Arabic handwritten recognition",Springer, pp. 235-244, 2010.

[Baudat et Anouar, 2000] G. Baudat et F. Anouar, "Generalized Discriminant Analysis Using a Kernel Approach", Neural Computation, 12(10) :2385-2404,2000.

[Belhumeur et al., 1997] Belhumeur, P. N., Hespanha, J. P., et Kriegman, D. J. 1997. "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection.", IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7), 711-720.

[Bellman, 1961] Bellman, R. 1961. Adaptative Control Processes: A Guided Tour.Princeton University Press, New Jersey.

[Ben Amara et al 2000] Ben Amara N., Belaïd A., Ellouze N. (2000), Utilisation des modèles markoviens en reconnaissance de l'écriture arabe état de l'art. In Colloque International Francophone sur l'Ecrit et le Document (CIFED'00), Lyon, France.

[Berrani, 2004] S.A. Berrani,. 2004. Recherche approximative de plus proches voisins avec contrôle probabiliste de la précision ; application à la recherche d'images par le contenu. Thèse de doctorat, Université de Rennes 1,

Thomson Multimédia R&D France / Projet TEXMEX (IRISA).

[Beyer et al., 1999] K. S Beyer, J. Goldstein, R.Ramakrishnan et U.Shaft, When Is "Nearest Neighbor" Meaningful? Pages 217-235 of: Proceedings of the International Conference on Database Theory (ICDT). Springer-Verlag. 1999..

[Blake et Merz, 1998] C. L. Blake and CJ. Merz.: UCI repository of machine learning databases, 1998.

[Blanz et al,1996] V. Blanz, B. Scholkopf, H.H. Bulthoff, e. Burges, V. Vapnik, et T. Vetter. "Comparison of view-based object recognition algorithms using realistic 3d models.", In ICANN, pages 251-256,1996.

[Boser et al, 1992] B. Boser, I. Guyon, et V. Vapnik. A training algorithm for optimal margin clas¬sifiers. In Fifth Annual Workshop on Computational Learning Theory, Pittsburg, 1992.

[Burges , 1998] C. Burges. A tutorial on support vector machine for pattern recognition, 1998.

[Burges et al, 1997] C. Burges et B. Scholkopf. Improving the accuracy and speed of support vector machines. In Michael e. Mozer, Michael I. Jordan, and Thomas Petsche, editors, Advances in Neural Information Processing Systems, volume 9, page 375. The MIT Press, 1997.

[Chaibi, 2013] S.Chaibi,: "A Hierarchical Classifier Using SVMs for Arabic Phonemes Recognition"; Int. Conf. On signal, Image, Vision and Their Application, SIVA2013, Guelma, Algerie, 451-457, 2013.

[Chaibi, 2013] S.Chaibi,: " SVM/K-Means Combination Proposition to Speeding up SVM Training"; Int. Conf. On signal, Image, Vision and Their Application, SIVA2013, Guelma, Algerie, 69-72, .2013.

[Chaibi et Laskri, 2014] **http://www.inderscience.com/info/ingeneral/forthcoming.php?jcode= IJBIDM**

[Chernoff, 1952] H. Chernoff, "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations". Annals of Mathematical Statistics 23 (4): 493–507. (1952).

[Comon, 1994] P. Comon, "Independent Component Analysis, a new concept for Signal Processing", Special issue on Higher-Order Statistics, 36(3), 287-314. 1994.

[Cortes et Vapnik, 1995] C. Cortes et V. Vapnik. Support-vector networks. Machine Learning, 20(3) :273¬297,1995.

[Courant et Hilbert, 1953] R.Courant et R Hilbert. "Methods of Mathematical Physics". Interscience, 1953.

[Demartines et érault, 1997] P. Demartines, et J. Hérault, "Curvilinear Component Analysis: a self-organising neural network for non-linear mapping of data sets". IEEE Transaction on Neural Networks, 1(8), 148-154. 1997.

[Dietterich et Bakiri, 1995] T. G. Dietterich et G. Bakiri (1995) "Solving Multiclass Learning Problems via Error-Correcting Output Codes", Volume 2, pages 263-286, Journal of Artificial Intelligence Research, 2 :263¬286,1995.

[Duda et al., 2000] R. O. Duda, P. E. Hart, et D. G. Stork, "Pattern Classification". John Wiley and Sons, Inc. 2000.

[Faloutsos et Lin, 1995] C. Faloutsos, et K.-I. Lin, "FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets." Pages 163-174 of: SIGMOD Conference. 1995.

[Farah et al, 2005] N. Farah, M.T. Khadir, et M. Sellami, "Artificial neural network fusion: application to Arabic word recognition", Proceedings of the European Symposium on Artificial Neural Networks (ESNN), Brugs-Belguim, pp. 27-29, 2005.

[Farah et al, 2005] N. Farah, L. Souici, et M. Sellami, "Arabic word recognition by classifiers and context", Journal of Computer Science and Technology , vol. 20, no. 3, pp.402-410, 2005.

[Fisher, 1936] R.A, Fisher. "The use of multiple measurements in taxonomie prob1ems", 1936.

[Freeman et Skapura, 1991] J.A. Freeman et D.M. Skapura. "Neural Networks Algorithms: Applications and Programming Techniques", Addison-Wesley Publishing Company, United States of America (USA), 1991.

[Gasmi et al,2005] I. Gasmi, H. Merouani, et L. Souici-Meslati, "Combinaison de classifieurs", Proceeding of the 3 rd International Conference : Sciences of Electronic, Technologies of Information and Telecommunication (SETIT), Sousse-Tunisia, 2005.

[Geman et al,1992] S. Geman, E. Bienenstock, et R Doursat. Neural networks and the bias/variance di1emma. Neural Computation, 4(1) :1-58,1992.

[Gharbi et al,2008] A. Gharbi et N. Farah, "Un système à base des SVM pour la reconnaissance d'images de mots des montants des chèques arabes manuscrits ", Proceedings du 4ème Symposium International sur les Images, Multimédias, applications Graphiques et Environnements, Guelma-Algeria, pp. 41-44, 2008.

[Gill et al, 1981] P. E. Gill, W. Murray, et M. H. Wright, *Practical Optimization*, Academic Press.

[Girosi, 1997] F. Girosi. "An equivalence between sparse approximation and support vector machines". Technica1 Report AIM-1606, 1997.

[Graupe, 2007] D. Graupe. "Principles of Artificial Neural Networks," World Scientific, Chicago-United States of America (USA), 2007

[Guermeur et Monfrini, 2011] Y. Guermeur, E. Monfrini, A Quadratic Loss Multi-Class SVM for which a Radius–Margin Bound Applies, INFORMATICA, 2011, Vol. 22, No. 1, 73–96.

[Guyon et Elisseeff, 2003] I. Guyon, et A. Elisseeff, "An introduction to variable and feature selection". Journal of Maching Learning Research, Special issue on variable and feature selection, 3, 1157-1182. 2003.

[Hassini et al, 2004] A. H. Hassini, X. L. Tang, J. F. Liu, et W. Zhao, "Printed Arabic character recognition using HMM", Jo u r n a l o f C o mp u t er S ci en ces &Technologies, vol. 19, no. 4, pp. 538-543, 2004.

[Hastie et Tibshirani,1998] T. Hastie et R. Tibshirani, Classification by pairwise coupling, Ann. Statist.Volume 26, Number 2 (1998), 451-471.

[Heisele et al,2001] B. Heisele, P. Ho, et T. Poggio, "Face recognition with support vector machines: global versus component-based approach", Proc. 8th International Conference on Computer Vision, vol. 2, pp. 688– 694, Vancouver, 2001.

[Hérault et al., 2002] J. Hérault, A. Guérin-Dugué, et P. Villemain, "Searching for the embedded manifolds in high-dimensional data, problems and unsolved questions". In: European Symposium on Artificial Neural Networks (ESANN). 2002.

[Hotelling, 1933] H. Hotelling, Analysis of complex statistical variables into principal components. Journal of Educational Psychology, 24, 417-441, 1933.

[Hsu etLin 2002] C. Hsu and C. Lin. A comparison of methods for multi-class support vector ma¬chines. IEEE Transactions on Neural Networks, 13 :415-425,2002.

[Jain et Zongker, 1997] A. Jain, and D. Zongker, "Feature Selection: Evaluation, Application, and Small Sample Performance." IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(2), 153-158. 1997.

[Joachims, 1999] T. Joachims. "Transductive inference for text classification using support vector machines", In Proc. 16th International Conference on Machine Learning, pages 200-209. Morgan Kaufmann, San Francisco, CA, 1999.

[Jutten, 1987] C. Jutten, "Calcul neuromimétique et traitement du signal - Analyse en composantes indépendantes. Thèse d'état, Institut National Polytechnique de Grenoble. 1987

[Keerthi et al, 1999] S. Keerthi, S. Shevade, C. Bhattacharyya, et K. Murthy. "Improvements to platt's smo algorithm for svm classifier design", 1999.

[Khorsheed M.S, 2003] M.S. Khorsheed, "Recognizing Arabic manuscripts using a single hidden Markov model", Pattern Recognition Letters, 24, pp. 2235-2242. (2003)

[Kolenda et al., 2002] T. Kolenda, , L. K. Hansen, J. Larsen, , et O. Winther, "Independent component analysis for understanding multimedia content". Pages 757-766 of: Bourlard, H., Adali, T., Bengio, S., Larsen, J., & Douglas, S. (eds), Proceedings of IEEE Workshop on Neural Networks for Signal Processing XII. IEEE Press. 2002.

[Kolmogorov,1965] A. Kolmogorov. "Three approaches to the quantitative definition of information". Problems Inform. Transmission, 1(1) :1-7,1965.

[Le Borgne et al., 2003] H. Le Borgne, Analyse de scènes naturelles par composantes indépendantes. Thèse de doctorat, Institut National Polytechnique de Grenoble. 2004.

[Lejsek et al., 2005] H.Lejsek, F.-H. Ásmundsson, B., Pór-Jónsson, et L.Amsaleg, Efficient

and Effective Image Copyright Enforcement. In: Journées Bases de Données Avancées BDA). 2005.

[MacQueen, 1967] J. B.MacQueen, "Some Methods for classification and Analysis of Multivariate Observations ". Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press. pp. 281–297. (1967).

[Martinez et Kak, 2001] J. Martinet, Y. Chiaramella, et P. A Mulhem, "Model for Weighting Image Objects in Home Photographs". Pages 760-767 of: ACM CIKM'05. 2005.

[Mika et al , 1999] S. Mika, G. Râtsch, J. Weston, B. Scholkopf, et K. Müller, "Fisher discriminant analysis with kernels", 1999.

[Mika, et al, 2001] S. Mika, G. Râtsch, et K-R. Müller. "A mathematical pro¬gramming approach to the kernel fisher algorithm". In NIPS, pages 591-597, 2000.

[Mika,et al, 2000] S. Mika, A. Smola, and B. Scholkopf. "An improved training algorithm for kernel fisher discriminants", 2001.

[Moddemeijer, 1989] R. Moddemeijer, "On Estimation of Entropy and Mutual Information of Continuous Distributions". Signal Processing, 16(3), 233-246. 1989.

[Nguyen et Phung, 2010] G. H. Nguyen, S.L. Phung,:" Efficient SVM Training with Reduced Weighted Samples"; World Congress on Computational Intelligence ,WCCI 2010, USA, 1764-1768.2010.

[Osuna et al, 1997] E. Osuna, R. Freund, et F. Girosi. "Improved Training Algorithm for Support Vector Machines", 1997.

[Osuna, et al 1997] E. Osuna, R. Freund, and F. Girosi. Improved training algorithm for support vector machines, 1997.

[Panda et al, 2006] N. Panda, E.Y. Chang, et G. Wu,: " Concept Boundary Detection For Speeding Up SVMs"; 23rd Int. Conf. on Machine learning(ICML '06) 2006, 681-688.2006.

[Platt et al, 1999] J.C. Platt. "Fast training of support vector machines using sequential minimal optimization". In B. Scholkopf, e. Burges, and A. Smola, editors, Advances in Kernel Methods - Support Vector Learning, chapter 12. 1999.

[Poggio., 1975] T. Poggio. "On optimal nonlinear associative recall". Biological

Cybernetics, 19 :201-209, 1975.

[Rissanen, 1978] J. Rissanen. "Modeling by shortest data description". Automatica, 14 :465-471, 1978.

[Roth et Steinhage., 1999] V. Roth et V. Steinhage. "Nonlinear discriminant analysis using kernel functions", Advances in Neural Information Processing Systems, 1999.

[Sari et Sellami, 2002] T. Sari et M. Sellami, "Morpho-lexical analysis for correcting arabic OCR-generated words", Proceedings of International Workshop on Frontiers Handwriting Recognition (IWFHR), Niagara on the lakes-Canada, pp. 461-466, 2002.

[Schmidt, 1996] M. Schmidt. "Identifying speakers with support vector networks", Proc. Interface, Sydney, 1996.

[Scholkopf et al, 1999] B. Scholkopf, C. Burges, et A. Smola, editors, Advances in Kernel Methods ¬Support Vector Learning, chapter 1. 1999.

[Schokopf et Smola., 2002] B. Scholkopf, O. Smola, R. Williams on, and P. Bartlett. New support vector algorithms. Neural Computation, 12 :1083-1121,2002.

[Smola et Scholkopf, 1998] A. Smola and B. Scholkopf. A tutorial on support vector regression, 1998.

[Smola, 1998] A. Smola. "Learning with Kernels". PhD thesis, GMD First, Berlin, Germany, 1998.

[Song & Fan, 2005] X.Song, et G.Fan, Key frame extraction for object-based video segmentation. In: IEEE Workshop on Motion and Video Computing. 2005.

[Souici et Sellami, 2002] L. Souici-Meslati et M. Sellami, "Reconnaissance de montants littéraux arabes par une approche hybride neuro-symbolique", Proceedings du 11 ème Congrès Francophone AFRIF-AFIA de Reconnaissance de Formes et Intelligence Artificielle (RFIA), Angers-France, pp. 917-926, 2002.

[Stitson, et al, 1997] M.O. Stitson, A. Gammerman, V. Vapnik, V.Vovk, C. Watkins, et 1. Weston. Support vector anova decomposition, 1997.

[Tax et Duin, 1999] D.M.J. Tax, et R.P.W Duin,., 1999. "Support Vector Domain Description." Pattern Recognition Letters 20. 1191-1199.

[Tenenbaum et al. J.B Tenenbaum,., V. de Silva, et J.C. Langford, (2000) A global

2000] network framework for nonlinear dimensionality reduction. *Science* 290, 2319–2323.

[Tipping ,2000] M. E. Tipping. "Sparse kemel principal component analysis". In NIPS, pages 633-639,2000.

[Torkkola et Campbell, 2000] K.Torkkola, et W. Campbell, "Mutual information in learning feature transformations". In: Proceedings of International Conference on Machine Learning. 2000.

[Vanderbei, 1999] R. J. Vanderbei. "Interior point methods : Algorithms and formulations". ORSA Journal ofComputing, 6(1) :32-34,1994.

[Vapnik, 1995] V. Vapnik. The Nature of Statistical Learning Theory. Springer Verlag, New York, 1995.

[Vapnik., 1979] V. Vapnik. Estimation of Dependences Based on Empirical Data in Russian. N auka, Moscow, Russia, 1979.

[Vapnik., 1982] V. Vapnik. Estimation of Dependences Based on Empirical Data. springer Verlag, Berlin, 1982.

[Vapnik., 1995] V. Vapnik. The Nature of Statistical Learning Theory. Springer Verlag, NY, USA, 1995.

[Vapnik., 1998] V. Vapnik. Statistical Learning Theory. NY, USA, 1998

[Vapnik., 1999] V.Vapnik., An overview of statisticalleaming theory. IEEE Transactions on Neural Ne tworks, 10(5), September 1999.

[Vapnik., 2000] V.Vapnik and Olivier Chapelle. Bounds on error expectation for support vector machines. Neural Computation, 12(9) :2013-2036,2000.

[Vasconcelos, 2003] N. Vasconcelos, "Feature selection by maximum marginal diversity: optimality and implications for visual recognition". Pages 762-769 of: proceedings of IEEE International Conference on Image Processing (ICIP), 2003.

[Wahba, 1998] G. Wahba. "Support vector machines, reproducing kemel hilbert spaces, and randomized gacv", 1998.

[Weber et al., 1998] R.Weber, H.-J, Schek, et S. Blott,. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. Pages 194-205 of: Proceedings of Internationale Conference of Very Large Data Bases (VLDB). 1998 (24-27).

[Weston, et al, 1997] A. Weston, M. Gammerman, V Vapnik, V Vovk, et C. Watkins. "Density estimation using support vector machines", 1997.

[Yang & Yang, 2003] J.Yang, et J.Y, Yang, "Why can LDA be performed in PCA transformed space?" Pattern Recognition, 36(2), 563-566. 2003.

[Zernike,1934] F. Zernike, "Diffraction theory of the knife-edge test and its improved version, the phase-contrast method," Physica 1, 689–704 (1934).

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

[Zuo et al., 2005] F. Zuo, P. H. N.With, et M. Veen, «Multistage Face recognition Using Adaptative Feature Selection and Classification.In: Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS). 2005.