République Algérienne Démocratique et Populaire

ministère de l'enseignement supérieur et la recherche scientifique

Université IBN KHALDOUN - Tiaret

faculté des sciences de l'ingénieur

département d'informatique

présente pour l'obtention de diplôme de

master en système embarqué

Type: Académique

Par: KEMOUNE NADIA

Thème

LA CLASSIFICATION DES DOCUMENTS PAR APPRENTISSAGE SVM

Dirigé par: Mr MOSTEFAOUI

année 2011/2012

Sample output to test PDF Combine only

Remerciements

A u terme de ce travail, je remercie ALLAH qui par sa grâce on est arrivé à ce stade.

J'aimerais tout spécialement remercier Monsieur S.MOSTEFAOUI pour m'avoir proposé un sujet de mémoire aussi passionnant, pour avoir renforcé mon intéret pour le Machine Learning au travers de ses explications, pour sa grande disponibilité , pour m'avoir toujours encouragé face à la difficulté, mais aussi pour sa gentillesse et ses conseils et aide qui ont bien contribué à son avancement. et surtout pour la confiance qu'il m'a accordé pour la réalisation de ce projet.

J'exprime toute ma gratitude à Monsieur HASSANI *et Monsieur* ALEM *qui m'ont fait l'honneur de participer à ma soutenance et d'avoir accepté de faire partie des membres du jury de cette thèse.*

Ce travail de recherche n'aurait jamais été terminé sans le soutien de plusieurs personnes qui n'ont jamais hésité à m'encourager durant toutes mes années. Je serais toujours incapable de retourner la faveur\l,a décision de faire une thèse a énormément pesé sur mon entourage. Je pense surtout à mes parents, mon frère et mes sœurs pour leur soutien inestimable, précieux et continue à tous les niveaux. C'est à eux que je dédie ce travail.

Je tiens à remercier infiniment l'ensemble des professeurs et le personnel du département informatique.

Pour terminer, j'aimerais remercier mes amis tout particulièrement M de longue date pour leur soutien, encouragements et confiance. A toute personne qui a apporté une quelconque aide aussi petite soit-elle lors de la réalisation de ce mémoire, qu'elle trouve ici l'expression de mon reconnaissance la plus chaleureuse.

Sample output to test PDF Combine only

Table des matières

Sample output to test PDF Combine only

Chapitre III :
Les algorithmes de décomposition

Chapitre IV:
Evaluation d'un classifieur,expérimentations, et résultats

Sample output to test PDF Combine only

Liste des figures

Sample output to test PDF Combine only

Liste des tableaux

Sample output to test PDF Combine only

INTRODUCTION

A l'heure actuelle, le nombre de sites en ligne, est estimé à 206, 026,787 selon Netcraft (http://news.netcraft.com/archives/web_server_survey.html selon leur dernière enquête en Mai 2010) dont la majorité du contenu textuel est écrite en plusieurs langues (anglais, français, russe, chinois, arabe, persan, hébreu, etc.). L'organisation de toute cette immense et gigantesque ressource est donc indispensable. Ainsi, les techniques de l'apprentissage artificiel, dont la classification automatique des documents fait parti, s'avèrent d'être pertinentes et très efficaces.

Depuis les années **1960**, les chercheurs se sont intéressés à la question de la classification automatique des documents. La majorité des travaux a été réalisée sur des documents écrits en caractères latins (français, anglais, espagnol, etc.). Mais les premiers pas dans ce domaine ont surtout été motivés par les progrès technologiques des années **1980** qui ont permis d'augmenter les capacités de stockage numérique et, par conséquent, le nombre de documents textuels à traiter. Grâce à ces progrès technologiques, le volume de documents numériques n'a cessé de croître jusqu'à en rendre impossible une classification manuelle. Des besoins en classification automatique se sont donc fait ressentir aussi bien sur Internet (moteurs de recherche) qu'au sein des entreprises (classement de documents internes, de dépêches d'agences, etc.).

Le choix de l'algorithme d'apprentissage se présente comme un autre élément essentiel pour une classification automatique efficace. La plupart des travaux menés sur des documents se basent sur des algorithmes d'apprentissage récents comme, par exemple, , les réseaux bayésiens naïfs, et les arbres de décision, les machines à vecteurs de support qui sont connus pour être parmi les plus performants classifieurs du domaine ,C'est dans ce cadre que se situe notre travail, qui vise à explorer le potentiel des techniques d'apprentissage ,d'une part, répondre aux besoins de la classification des documents et d'autre part présenter plusieurs avantages en matière des gains en complexité algorithmique et en espace mémoire.

Sample output to test PDF Combine only

I.1) Introduction

L'apprentissage artificiel ou automatique (en anglais, « *Machine Learning »))* selon **[Mitchell, 1997]** est défini comme,

> *« Un sous-domaine de l'intelligence artificielle qui s'intéresse à conférer aux machines la capacité de s'améliorer à l'accomplissement d'une tâche, en interagissant avec leur environnement. »*

-**L**'apprentissage artificiel se divise principalement en deux façons : l'apprentissage *supervisé* et l'apprentissage *non-supervisé*.

- **L**a classification (ou catégorisation[1]) automatique des documents d'une manière supervisée est le processus d'assigner d'une façon autonome et automatique des documents à une ou plusieurs catégories prédéfinies (ex. Politique, Economie, Sport, etc.).
- **L**a manière dite non-supervisée, quant à elle, ignore les catégories de sortie et c'est à l'algorithme d'apprentissage d'analyser les documents pour les concevoir.

C'est dans l'approche dite supervisée que s'inscrit la façon dont on aborde aujourd'hui le problème de la catégorisation automatique de documents.

I.2) Définition de la classification automatique des documents : quoi et comment ?

Le but de la catégorisation automatique de textes est d'apprendre à une machine à classer un texte dans la bonne catégorie en se basant sur son contenu. Habituellement, les catégories font référence aux sujets des textes, mais pour des applications particulières, elles peuvent prendre d'autres formes. En effet, on peut résoudre, par des techniques de catégorisation, des problèmes tels que l'identification de la langue d'un document, le filtrage du courrier électronique pertinent ou indésirable, ou encore la désambiguïsation de termes**.** Un autre

[1] Dans ce travail, les deux termes *classification* et *catégorisation* sont interchangeables et désignent le même concept.

Sample output to test PDF Combine only

aspect du problème qui varie selon les applications est la présence ou non d'une contrainte concernant le nombre de catégories assignables à un document donné. Il se peut qu'on désire qu'un même texte ne soit associé qu'à une seule catégorie ou bien on peut permettre que plusieurs catégories accueillent un même document. Aussi, une précision supplémentaire est à faire : dans le cadre de la catégorisation de textes, l'ensemble de catégories possibles est déterminé à l'avance. Il est à noter que le problème consiste à regrouper des documents selon leur similarité.

Dans une catégorisation de texte : la classification s'apparente au problème de l'extraction de la sémantique d'un texte, puisque l'appartenance d'un document à une catégorie est étroitement liée à la signification de ce texte. C'est en partie ce qui rend la tâche difficile puisque le traitement de la sémantique d'un document écrit en langage naturel n'est pas encore solutionné. Une observation mérite aussi d'être faite concernant le fait que la nature des textes à traiter influence significativement la difficulté de la tâche de classification. Prenons l'exemple d'articles de journaux écrits généralement dans un style direct et contenant de l'information purement factuelle. Le vocabulaire utilisé s'avère précis et souvent relativement restreint pour faciliter la compréhension. A l'opposé, imaginons un corpus de textes d'un style plus littéraire, utilisant un vocabulaire très varié et imagé. On peut aisément prévoir que la classification automatique de ce dernier corpus présentera plus de difficultés que pour l'autre. Entre ces deux extrêmes, on peut aussi retrouver des textes scientifiques (où chaque catégorie aura potentiellement un vocabulaire caractéristique), des pages Web, du courrier électronique, etc. Chacun de ces types de textes possède des particularités qui rendent la tâche de classification plus ou moins ardue. **[Sim, 2005].**

Selon **[Sebastiani, 2002]** page 3, il y a deux observations fondamentales à respecter pour obtenir une catégorisation généralisable :

- *« Les catégories ne sont que des labels symboliques. Ce qui veut dire que le sens ou la signification du nom de la catégorie n'a rien à voir avec la construction du classifieur. Autrement dit, on ne peut jamais utiliser le texte qui constitue le label (*par exemple, Economie ou Sports*) dans le processus de la catégorisation.*

- *L'affectation d'un document à une catégorie doit, en général, se baser sur le contenu du document plutôt que ses métadonnées (par exemple, date de*

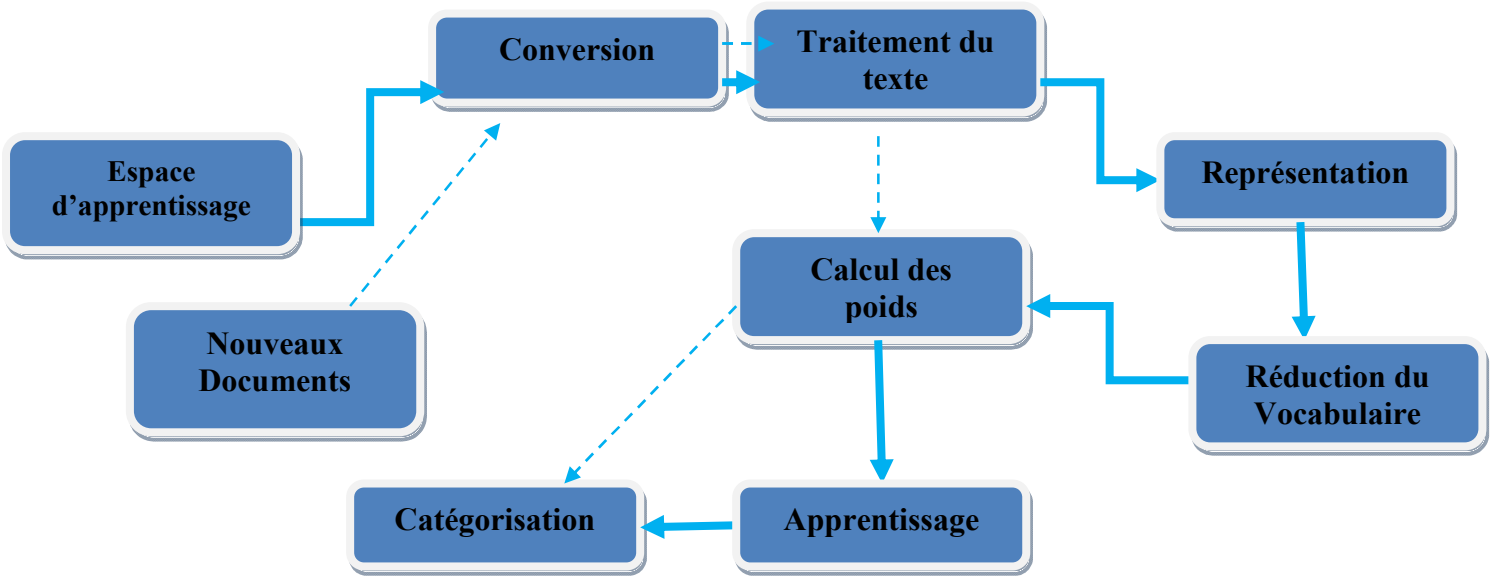
Sample output to test PDF Combine only

publication, le type de document, le nom de l'auteur, etc.). »

I.2.1) La similarité textuelle :

La notion de similarité textuelle est très souvent utilisée dans les applications de Traitement de la langue destinées à l'exploitation de corpus textuels de grande taille. Par exemple, en Recherched ocumentaire, les documents pertinents retournés par le moteur de recherche sont les plus proches de la requête selon une certaine mesure de similarité **[Salton et McGill, 1983]** ; de même, dans le cas de la structuration automatique de bases de données textuelles (classification non supervisée), les documents sont également regroupés en classes en fonction d'une mesure de similarité spécifique **[Salton et al, 1975]**.

L'objectif de nombreuses tâches des systèmes de traitement du langage naturel comme la recherche documentaire (RD) et la classification automatique, repose sur la notion de similarité textuelle qui est généralement calculée à partir d'une représentation spécifique des documents dans un espace défini à priori. Une approche standard très utilisée en RD, est l'utilisation de mesures de similarité (au sens mathématique) sur des représentations vectorielles des documents.



FGI 1.1 Le schéma général de la classification automatique des documents

Sample output to test PDF Combine only

I.3) Comment représenter un document ?

« Puisqu'il n'existe pas actuellement une méthode d'apprentissage capable d'exploiter un texte directement sans que ce dernier soit bien structuré, une transformation préliminaire est alors indispensable » **[Sebastiani, 2002]**.

Il est souvent admis que le sens d'un document peut être porté par un ensemble d'unités linguistiques particulières, caractéristiques plus ou moins élaborées issues de l'analyse du corpus documentaire **[Besançon, 2002].** Chaque unité linguistique définit une « idée », et par conséquent, un ensemble de mêmes unités linguistiques est censé définir un sens identique.

Les premières unités linguistiques à s'être imposées comme représentatives du sens sont le radical et le lemme des mots. La reconnaissance de ces unités linguistiques nécessite d'effectuer un prétraitement linguistique des mots du texte. Certaines unités linguistiques, plus rudimentaires, ne nécessitent aucun traitement a priori, comme le « sac de mots » ou la phrase.

Pour la majorité des méthodes d'apprentissage, il faut transformer l'ensemble des textes en un tableau croisé « Individus-Variables » :

- L'individu est un texte (un document) d_j, étiqueté lors de la phase d'apprentissage, il est classé dans la phase de prédiction.

- Les variables sont les descripteurs (les termes) t_k qui sont extraits des données textuelles.

I.3.1) Approches pour la représentation des textes :

a) Le mot « sac de mots » :

La représentation de textes la plus simple a été introduite dans le cadre du modèle vectoriel elle porte le nom de « sac de mots ». L'idée est de transformer les textes en vecteurs dont chaque composante représente un mot. Dans cette approche l'unité linguistique choisie est le mot tel qu'il apparaît dans le document Cette approche ne nécessite aucun traitement préliminaire, chaque mot est extrait du texte en considérant des séparateurs entre chaque mot, comme l'espace, la virgule, la tabulation, le point et la ponctuation en général. L'approche par sac de mots est généralement associée à une procédure de filtrage. En effet, le nombre de mots caractérisant un corpus de documents peut rapidement atteindre une centaine de milliers. Il devient alors nécessaire, afin d'envisager un traitement analytique des textes, de ne

Sample output to test PDF Combine only

conserver qu'un sous-ensemble de ces mots. Classiquement, le filtrage repose à la base sur les fréquences d'occurrence des mots dans le corpus.

Il est assez courant de définir une liste de rejet (*stop words*) de mots à ne pas considérer, tels les pronoms, les articles, etc...

Comme nous l'avons déjà indiqué avant il n'existe pas à l'heure actuelle une méthode d'apprentissage artificiel capable d'exploiter directement les documents de l'espace d'apprentissage dans leur état d'origine. Ces derniers doivent être représentés sous une autre forme qui permet à l'algorithme d'apprentissage de les utiliser. Le meilleur choix de la représentation des ces documents reste toujours un sujet de recherche et de débat.
L'approche de représentation adoptée dans cette thèse, et celle souvent utilisée dans la littérature« *sac-de-mots* » (en anglais, *«bag-of-words»)*. Selon cette approche, on construit un tableau croisé « documents × attributs » où les lignes correspondent aux documents à catégoriser et les colonnes correspondent à leurs *attributs*. Le choix de la nature de l'attribut d'un document est l'enjeu principal d'une classification automatique efficace.

Dans cette thèse nous adoptons les racines comme attributs puisqu'elles ont contribuées à la meilleure performance de classification. Ce choix sera validé pour construire le tableau croisé issue de l'approche « sac de mots », nous commençons par construire un vecteur global $\boldsymbol{V}$ de taille $|\boldsymbol{T}|$ contenant l'intégralité des attributs $\{t1, t2,\ldots, t|T|\}$ du corpus, où $|\boldsymbol{T}|$ est la taille du corpus d'apprentissage le nombre total des attributs *distincts* qui apparaissent au moins une seul fois dans les documents.

$$V = \{t_1, t_2,\ldots, t_{|T|}\}$$

Ensuite, chaque document d_i est transformé en un vecteur v_i composé de |T| entrées f_{ki} qui représentent les fréquences des attributs qui constituent le document d_i.

$$v_i = \{f_{1i}, f_{2i},\ldots, f_{|T|i}\}$$

Ces vecteurs seront ensuite repartis dans un tableau croisé où chaque fréquence sera remplacée par un poids p_{ki}

Pour mieux comprendre cette approche, considérons l'exemple suivant :

D_1 = Saeed aime regarder la nature. Wafaa aime la regarder aussi.

D_2 = Saeed aime aussi regarder les matches de foot.

Basé sur ces deux documents on construit le vecteur global suivant :
V ={1 : « Saeed », 2 : « aime », 3 : « regarder », 4 : « la », 5 : « nature », 6 : « aussi », 7 : « les », 8 : « matches », 9 : « de », 10 : « foot », 11 : «Wafaa »} contenant 11 mots

Sample output to test PDF Combine only

distincts et donc |T| = 11. En utilisant ce vecteur, nous représentons chaque document par un vecteur v_i composé de 11 entrées :

$$v_1 = \{1, 2, 2, 2, 1, 1,0,0,0,0,1 \}$$

$$v_2 = \{1, 1, 1, 0, 0, 0,1,1,1,1,0 \}$$

Chacune de ces entrées représente la fréquence de chaque attribut dans le document qui le contient. Ces vecteurs sont ensuite repartis dans le tableau croisé suivant :

Mots / Document	Saeed	aime	regarder	la	nature	les	smatche	ed	foot	wafaa	aussi
D_1	1	2	2	2	1	1	0	0	0	0	1
D_2	1	1	1	0	0	0	1	1	1	1	0

Tab I.1 Un exemple d'un tableau croisé

b) des phrases :

Malgré la simplicité de l'utilisation de mots comme unité de représentation, certains auteurs proposent plutôt d'utiliser les phrases comme unité. Les phrases sont plus informatives que les mots seuls, car les phrases ont l'avantage de conserver l'information relative à la position du mot dans la phrase.

Logiquement, une telle représentation doit obtenir de meilleurs résultats que ceux obtenus via les mots. Mais les expériences présentées ne sont pas concluantes car, si les qualités sémantiques sont conservées, les qualités statistiques sont largement dégradées. **[Rad, 2003].**

I.3.2) Le prétraitement des documents

I.3.2.1) Stemmatisation (radicalisation ou « stemming »)

L'utilisation de mots comme unité linguistique est possible, mais pose toutefois un certain nombre de problèmes. En effet, il existe plusieurs centaines de milliers de mots dans un corpus documentaire et associer un sens à chacun de ces mots n'est pas nécessairement des plus pertinents. En effet, beaucoup de mots ont des racines communes, des sens communs.

Attribuer un sens différent à « chantaient » et « chantent » par exemple, relèverait d'une redondance sans pertinence sémantique.

Sample output to test PDF Combine only

La stemmatisation qui se nomme également radicalisation ou en anglais « stemming » est un prétraitement qui consiste à trouver la racine de chaque mot. C'est un traitement qui procède à une analyse morphologique du texte. Une implémentation très connue est celle de Porter **[Porter, 1980].** Cet algorithme consiste principalement à effectuer une « désuffixation ». C'est un traitement fondé sur l'étude de la morphologie des mots. Il se base sur un dictionnaire de suffixes qui permet d'extraire le radical du mot.

La stemmatisation de : « Ces vases sont disproportionnés », grâce à l'algorithme de Porter donne : « Ce vas sont disproportion ». Une vraie radicalisation aurait donné : « Ce vase sont proportion » **[Plantié, 2006]**.

I.3.2.2) Lemmatisation

La lemmatisation nécessite une analyse plus poussée que la stemmatisation. La lemmatisation se fonde sur un lexique qui est un ensemble de lemmes, que l'on peut assimiler globalement aux entrées d'un dictionnaire.

L'objectif de la lemmatisation est d'associer à chaque mot une entrée dans le lexique. Or, l'analyse morphologique est insuffisante pour extraire les lemmes d'un texte car de nombreux mots de même graphie peuvent provenir de différents lemmes. Par exemple « offense » peut être considéré comme le lemme définissant la parole ou la faute qui blesse, ou comme le verbe "offenser" conjugué. Cette ambiguïté se résoud en analysant la catégorie grammaticale du mot en question dans la phrase. La lemmatisation nécessite donc de réaliser une analyse supplémentaire : l'analyse syntaxique. La lemmatisation recouvre donc deux analyses regroupées sous le terme d'analyse morphosyntaxique. Depuis la fin des années 80, les lemmatiseurs sont capables d'associer à chaque mot d'un texte, son lemme, grâce à un étiqueteur morphosyntaxique (nom, verbe, adjectif, etc.) dont les taux de réussite avoisinent les 90% **[Schmid, 1994].** La construction d'un lemmatiseur nécessite néanmoins un étiquetage de quelques milliers de mots.

I.3.2.3) La « stop-list »

Avant d'effectuer un des prétraitements précédents, il est usuel d'utiliser une «stop-list ». L'objectif de la « stop-list » est d'éliminer tous les mots ne participant pas activement au sens du document. La « stop-list » est une liste répertoriant tous les mots outils (pronoms, articles,

Sample output to test PDF Combine only

etc.) et les mots trop fréquents pour être discriminants. D'un point de vue linguistique les mots outils sont par définition des mots "vides" de sens. Et, d'un point de vue statistique, les mots trop fréquents (et de distribution uniforme) ne sont d'aucune aide à un processus de catégorisation puisque non discriminants. De par sa nature ce prétraitement s'effectue donc en amont des autres prétraitements linguistiques et son objectif est d'éliminer toutes les unités linguistiques non discriminantes. L'inconvénient de la « stop-list » est qu'elle reste dépendante d'une langue donnée. La « stop-list » pose un autre problème plus fondamental.

En effet, qui peut dire a priori que tel type syntaxique ou mot « outil » est vide de sens ? On risque dans certains cas d'omettre certaines unités linguistiques qui auraient permis d'apporter une aide précieuse à la tâche de classification. En particulier, il est peut être imprudent d'établir cette liste avant d'avoir fait le choix de la technique de classification qui peut parfois nécessiter toutes les informations disponibles.

I.4) Les modèles vectoriels de représentation de documents :

Un **modèle vectoriel** (parfois nommé **sémantique vectorielle**) est une méthode algébrique de représentation d'un document visant à rendre compte de sémantique. Elle est utilisée en recherche d'information, notamment pour la recherche documentaire, la classification ou le filtrage de données. Ce modèle concernait originellement les documents textuels et a été étendu depuis à d'autres types de contenus.

Le modèle vectoriel est une représentation mathématique du contenu d'un document, selon une approche algébrique.

L'ensemble de représentation des documents est un vocabulaire comprenant des termes d'indexation. Ceux-ci sont typiquement les mots les plus significatifs du corpus considéré: noms communs, noms propres, adjectifs... Éventuellement ils peuvent être des constructions plus élaborées comme des expressions ou des entités sémantiques). À chaque élément du vocabulaire est associé un index unique arbitraire.

Chaque contenu est ainsi représenté par un vecteur v, dont la dimension correspond à la taille du vocabulaire. Chaque élément v_i du vecteur v consiste en un poids associé au terme d'indice i et à l'échantillon de texte. Un exemple simple est d'identifier v_i au nombre d'occurrences du terme i dans l'échantillon de texte. La composante du vecteur représente donc le poids du mot i dans le document.

Sample output to test PDF Combine only

Deux textes utilisant les mêmes segments textuels seront donc projetés sur des vecteurs identiques. Le formalisme le plus utilisé pour représenter les textes est le formalisme vectoriel issu de **[Salton, 1971]** et **[Salton, 1983]**.

oCnoisulcn

Ce chapitre introductif a élaboré et défini l'apprentissage artificiel et la classification automatique des documents ainsi que son objectif qui sont devenus des aspects majeurs et distingués dans le domaine de la recherche d'information. Nous avons présenté, au premier temps ,la notion de la similarité textuelle ainsi les différents modèles de représentation de textes permettant de calculer cette similarité sémantique entre les documents. nous avons introduit le modèle vectoriel, qui tentent de prendre en compte la structure sémantique des unités linguistiques.

Dans ce qui suit, nous présentons une description générale des algorithmes d'apprentissage utilisés dans les travaux menés dans cette thèse.

Sample output to test PDF Combine only

II.1) Introduction

En apprentissage automatique, différents types de classificateurs ont été mis au point, et cela dont le but d'atteindre un degré maximal de précision et d'efficacité, chacun ayant ses avantages et ses inconvénients. Mais, ils partagent toute fois des caractéristiques communes.

Parmi la panoplie de classificateurs existants, on peut faire des regroupements et distinguer des grandes familles.

Dans les pages qui suivent, nous allons exposer quelques algorithmes en détail, le classificateur bayésien naïf algorithme surpassé par d'autres mais il est souvent utilisé comme point de référence à cause de sa simplicité, l'algorithme Arbres de décision *(AD)*. Puis, les machines à support vectoriel, qui représentent vraisemblablement à l'heure actuelle le meilleur choix en catégorisation de textes.

II.2) Formulation de la classification en théorie :

La tâche qu'un classificateur doit effectuer peut être exprimée par une fonction que l'on appelle fonction de décision : $f : X \rightarrow Y$

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 le cas de la classification binaire, l'ensemble Y correspond à $\{+1, -1\}$. 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.

II.3) Algorithmes de classification

II.3.1) Naïve Bayes

Comme son nom l'indique, ce classificateur se base sur le théorème de Bayes permettant de calculer les probabilités conditionnelles. Dans un contexte général, ce théorème fournit une façon de calculer la probabilité conditionnelle d'une cause sachant la présence d'un effet, à partir de la probabilité conditionnelle de l'effet sachant la présence de la cause ainsi que des

Sample output to test PDF Combine only

probabilités a priori de la cause et de l'effet.

On peut résumer son utilisation lorsqu'il est appliqué à la classification de textes ainsi :

- On cherche la classification qui maximise la probabilité d'observer les mots du document.

- Lors de la phase d'entraînement, le classificateur calcule les probabilités qu'un nouveau document appartienne à telle catégorie à partir de la proportion des documents d'entraînement appartenant à cette catégorie. Il calcule aussi la probabilité qu'un mot donné soit présent dans un texte, sachant que ce texte appartient à telle catégorie.

- **P**ar la suite, quant un nouveau document doit être classé, on calcule les probabilités qu'il appartienne à chacune des catégories à l'aide de la règle de Bayes et des chiffres calculés à l'étape précédente.

La probabilité à estimer est donc : P($c_j \mid a_1 , a_2 , a_3 , ..., a_n$)

Où:

· c_j est une catégorie

· a_i est un attribut

A l'aide du théorème de Bayes, on obtient : $\frac{P(a_1,a_2,a_3,...,a_n \setminus c_j)P(c_j)}{P(a_1,a_2,a_3,...,a_n)}$

On peut omettre de calculer le dénominateur, qui reste le même pour chaque catégorie

En guise de simplification, on calcule P ($a_1, a_2, a_3, ..., a_n \mid c_j$) ainsi : $\prod_{i=1}^{n} P(a_i \setminus c_j)$

La probabilité qu'un mot apparaisse dans un texte est indépendante de la présence des autres mots du texte. On sait que cela est faux. Par exemple, la probabilité de présence du mot *«artificielle»* dépend partiellement de la présence du mot *«intelligence»*. Pourtant, cette supposition n'empêche pas un tel classificateur de présenter des résultats satisfaisants. Et surtout, elle réduit de beaucoup les calculs nécessaires. Sans elle, il faudrait tenir compte de toutes les combinaisons possibles de mots dans un texte, ce qui d'une part impliquerait un nombre important de calculs, mais aussi réduirait la qualité statistique de l'estimation, puisque la fréquence d'apparition de chacune des combinaisons serait très inférieure à la fréquence d'apparition des mots pris séparément. **[Sim, 2005].**

Pour estimer la probabilité P ($a_i \mid c_j$), on pourrait calculer directement dans les documents d'entraînement la proportion de ceux appartenant à la classe c_j qui contiennent le mot a_i.

Sample output to test PDF Combine only

Cependant, l'estimation ne serait pas très valide pour des numérateurs petits.

Dans le cas extrême où un mot ne serait pas du tout rencontré dans une classe, sa probabilité de 0 dominerait les autres dans le produit ci-dessus et rendrait nulle la probabilité globale. Pour pallier ce problème, une bonne façon de faire est d'utiliser le m-estimé qui est calculé ainsi :

$$\frac{n_k + 1}{n + |vocabulaire|}$$

Où

· n_k est le nombre d'occurrences du terme dans la classe c_j

· n est le compte total des mots dans le corpus d'entraînement.

· |Vocabulaire| : Le nombre de mots clés.

II.3.2) Arbres de décision (AD)

Les arbres de décision sont une des techniques les plus populaires du ML supervisé. Cette méthode est très facile à mettre en oeuvre, de plus, on peut facilement interpréter les règles de décision issues de l'apprentissage. Il existe plusieurs versions des AD dont les plus connues sont ID3 **[J. R. Quinlan,1993]** et C4.5 [**J. R. Quinlan,1996**]. Nous présentons ici la version la plus simple : ID3. Nous considérons à nouveau la classification de textes représentés par *bags of word* avec des poids binaires.

Un AD est constitué, comme tout arbre, de noeuds et de feuilles. Chaque nœud contient un test et possède autant de descendants qu'il y a de valeurs possibles pour ce test. Dans notre cas, les tests consisteront simplement à vérifier la présence d'un terme dans le document analysé. On aura donc affaire à des arbres binaires. Aux extrémités des branches de l'arbre, on trouve des feuilles qui indiquent le résultat de la classification, c.à.d la classe que l'on assigne. Pour classer un nouveau document, il suffit de le soumettre à la racine de l'arbre et de le laisser parcourir les branches au fil des résultats des tests jusqu'à atteindre une feuille.

Dans le cadre des AD, l'entraînement consiste à créer l'arbre à partir du *training set*. Il s'agit d'un processus récursif dans lequel les données d'entraînement seront utilisées pour déterminer l'attribut à tester lors de la création des noeuds. Au départ, on dispose d'un ensemble de documents correspondant au training set complet. Sur base de cet ensemble, on

Sample output to test PDF Combine only

va déterminer un attribut (dans notre cas, il s'agit de l'indice d'un terme) sur lequel le test de la racine de l'arbre va porter. Notre ensemble va ensuite être divisé en deux sous-ensembles contenant respectivement les données ayant et n'ayant pas satisfait au test. Le processus est alors répété sur ces deux sous-ensembles. Lorsque l'on ne peut plus trouver d'attribut permettant de subdiviser l'ensemble, une feuille portant le label majoritairement représenté par les exemples que l'on a reçus, est créée.

Il existe plusieurs manières de déterminer les tests à réaliser dans les noeuds. L'idée commune est de créer des tests qui discriminent le plus possible les exemples d'apprentissage. **E**n clair, cela veut dire que l'attribut sur lequel on fait le test doit séparer les exemples en deux ensembles de taille voisine. Dans ID3, on se sert d'un critère basé sur l'entropie, notion issue de la théorie de l'information. L'entropie du *training set* se définit comme suit :

$$Entropie(Tr) = \sum_{y \in Y} P(Y = y) \log_2 P(Y = y)$$

L'entropie permet de mesurer l'homogénéité des exemples. Si l'entropie vaut 0, tous les exemples appartiennent à la même catégorie, alors que si elle vaut 1, il y a autant d'exemples positifs que négatifs. Notons par ailleurs que dans ce contexte on définit 0 log 0 = 0.

L'entropie est utilisée pour formuler une mesure appelée Information Gain (IG) :

$$IG(Tr, j) = Entropie(Tr) - \left(\frac{|Tr_{j,1}|}{|Tr|} * Entropie(Tr_{j,1}) \right) - \left(\frac{|Tr_{j,0}|}{|Tr|} * Entropie(Tr_{j,0}) \right)$$

Où $Tr_{j,k}=\{(x,y) \in Tr \mid w_j=k\}$ et j est l'index de l'attribut (le terme) pour lequel le gain d'information est calculé.

Cette mesure est d'autant plus grande que l'attribut j est discriminant. En conséquence de quoi, lors de la création des noeuds, l'index de l'attribut sur lequel portera le test sera déterminé de la façon suivante :

$$best = \arg\max_j (IG(Tr', j))$$

Où Tr′ est le sous-ensemble des exemples du *training set* ayant satisfait aux tests menant au noeud courant.

Sample output to test PDF Combine only

II.4) Machines à Vecteurs Supports (SVM):

SVM est une méthode de classification qui fut introduite par **[Vapnik, 1995]**. Le succès de cette méthode est justifié par les solides bases théoriques qui la soutiennent. Il existe en effet un lien direct entre la théorie de l'apprentissage statistique et l'algorithme d'apprentissage de SVM. La plupart des techniques du ML possèdent un (trop) grand nombre de paramètres d'apprentissage à fixer par l'utilisateur (structure d'un réseau de neurones, coefficient de mise à jour du gradient, . . .). De plus, avec ces méthodes, le nombre de paramètres à calculer par l'algorithme d'apprentissage est en relation linéaire, voire exponentielle, avec la dimension de l'espace d'entrée. La formulation élégante de SVM laisse très peu de place aux paramètres utilisateurs et le nombre de paramètres est linéaire en la taille du *training set*. SVM est donc une méthode de classification particulièrement bien adaptée pour traiter des données de très haute dimension telles que les textes et les images. Dans la suite nous présentons les aspects théoriques de la méthode SVM.

II.4.1) Qu'est-ce qu'un SVM?

Une SVM est un algorithme d'apprentissage, permettant d'apprendre un séparateur. Ceci ramène le problème à savoir ce qu'est un séparateur. Donnons-nous un ensemble fini de vecteurs de R^n, séparés en deux groupes, ou dit autrement en deux classes. L'appartenance à un groupe ou un autre est défini par une étiquette, associée à chacun des vecteurs, sur laquelle est inscrite « groupe 1 » ou « groupe 2 ». Trouver un séparateur revient à construire une fonction, qui prend un vecteur de notre ensemble, et peut dire de quel groupe il est. Les SVM sont une solution à ce problème, comme le serait un simple apprentissage par coeur des classes associées aux vecteurs de notre ensemble. Mais avec les SVM, on attend de bonne propriétés de généralisation, à savoir que si un nouveau vecteur de présente, qui n'était pas dans l'ensemble, la SVM saura dire à quel groupe il est vraisemblable qu'il appartient, au regard des attributions de classes des vecteurs présents au départ.

L'idée est de poser, à partir des vecteurs dont on connaît les classes, un problème d'optimisation, du style « optimiser telle grandeur en s'assurant que ... ». Il y a deux difficultés. La première, c'est poser le bon problème d'optimisation. Cette notion de « bon » problème est celle qui fait référence aux théories mathématiques de la généralisation, et fait des SVM un outil d'un abord parfois difficile. La deuxième difficulté est de résoudre ce

Sample output to test PDF Combine only

problème d'optimisation une fois posé, et là on tombe plus dans des subtilités informatiques, dont nous retiendrons l'algorithme SMO.

II.4.2) Classificateur linéaire

II.4.2.1) Définition

Un classificateur est dit linéaire lorsqu'il est possible d'exprimer sa fonction de décision par une fonction linéaire en x qui désignera un vecteur de R^n. où n est le nombre de composantes des vecteurs contenant les données. On peut, en toute généralité, exprimer une telle fonction comme ceci : $f(x) = \langle w, x \rangle + b = \sum_{i=1}^{n} w_i x_i + b$

Où $w \in R^n$ et $b \in R$ sont des paramètres, et $x \in R^n$ est une variable.

Ce classificateur ne fournit pas des valeurs valant exclusivement -1 *(classe 1)* ou 1 *(classe 2*), mais nous dirons que quand le résultat $f_{w,b}(x)$ est positif, le vecteur x appartient à la même classe que les exemples d'étiquette 1, et que quand ce résultat est négatif, le vecteur x appartient à la même classe que les exemples d'étiquette -1.

Notons que l'équation $f_{w,b}(x)=0$ définit la frontière de séparation entre les deux classes, et que cette frontière est un hyperplan affine dans le cas du séparateur linéaire. En orientant l'hyperplan (C.à.d, en fixant un côté pour lequel les exemples sont classés positivement), la règle de décision correspond à observer de quel coté de l'hyperplan se trouve l'exemple x. La figure 4.1 représente la situation dans R^2. On voit que le vecteur w définit la pente de l'hyperplan : w est perpendiculaire à l'hyperplan. Le terme b quant à lui permet de translater l'hyperplan parallèlement à lui-même.

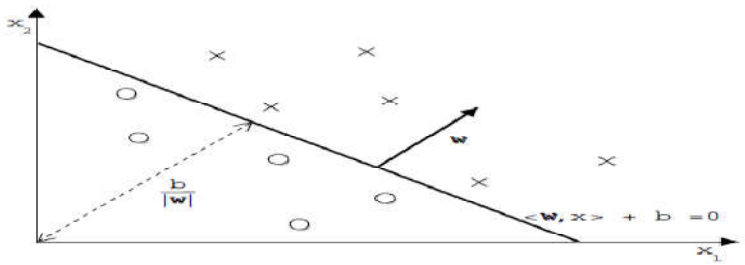


FIG.II.1 : Représentation dans R^2 de l'hyperplan correspondant à la fonction de décision d'un classificateur linéaire

Sample output to test PDF Combine only

II.4.3) Marge de l'hyperplan

II.4.3.1) La séparabilité

Soit S= $\{(x_i, y_i) \in R^n \text{ x } \{-1,1\}\}$, i=1,…, n, l'ensemble des données d'apprentissage (*training set*) avec leur étiquette $y_i \in \{-1,1\}$, $S^+ = \{x/ (x,y) \in S \text{ et } y=1\}$ et $S^- = \{x/ (x,y) \in S \text{ et } y=-1\}$. Dire que S est linéairement séparable signifie qu'il existe $w \in R^n$ et b un réel tel que :

$$f_{w,b}(x) = \langle w, x \rangle + b > 0 \qquad \forall x \in S^+$$

$$f_{w,b}(x) = \langle w, x \rangle + b < 0 \qquad \forall x \in S^-$$

Autrement dit, un *training set* S est linéairement séparable SSI :

$$\exists w \in R^n, b \in R : y_i(\langle w, x_i \rangle + b) \geq 0 \qquad \forall i = 1...n$$

La définition consiste à dire qu'il doit exister un hyperplan laissant d'un coté toutes les données positives et de l'autre, toutes les données négatives. Dans le cas de données linéairement séparables, il existe plusieurs méthodes pour trouver un tel hyperplan. La première d'entre elles et la plus connue est l'algorithme du perceptron de **[Rosenblatt, 1958]**.

II.5.3.2) Marge

Supposons que S soit linéairement séparable pour ce qui suit. C'est une hypothèse forte, que l'on réduira par la suite, mais qui va nous permettre de poser quelques notions. L'idée au coeur des SVM est que, certes, il convient de séparer les exemples de chaque classe, mais qu'il faut que l'hyperplan passe « bien au milieu ». C'est pour définir cette notion de « bien au milieu » (**FIG.II.2**) que l'on introduit la marge.

FIG.II.2 : Les mêmes exemples sont à gauche comme à droite, séparés par une droite.

Sample output to test PDF Combine only

Aidons nous de la figure II.5.3.2.b pour faire les observations suivantes. Notons déjà que les courbes d'équation $f_{w,b}(x) = C$ sont des hyperplans parallèles, et que w est normal à ces hyperplans. Le paramètre b traduit un décalage de l'hyperplan séparateur, soit une translation des valeurs de $f_{w,b}$. La norme $||w||$ de w influence les courbes de niveau $f_{w,b}$=C. Plus la $||w||$ est élevée, plus les courbes de niveau sont serrées.

Si l'on souhaite une frontière séparatrice donnée, on se trouve confronté à une indétermination pour le choix de w et de b. N'importe quel vecteur w non nul perpendiculaire à l'hyperplan convient. Une fois choisi, on détermine b tel que b/ ||w|| soit la mesure orientée[1] de la distance de l'origine au plan de séparation.

La notion de marge peut être relative à un exemple particulier ou à l'ensemble du *training set*. De plus, on considère deux types de marges : fonctionnelle et géométrique.

II.4.3.3) L'hyperplan canonique

Dans le cadre de classificateurs à marge maximale, l'hyperplan séparateur correspond à la médiatrice du plus petit segment de droite reliant les enveloppes convexes des deux catégories. Notons que l'on suppose aussi que le *training set* est linéairement séparable. Dès lors, on peut définir deux plans se trouvant de part et d'autre de l'hyperplan et parallèles à celui-ci, sur lesquels reposent les exemples le plus proches. La figure (**FIG.II.4)** illustre cette situation.

Il est donc possible de redimensionner w et b de telle sorte que les deux plans parallèles aient respectivement pour équation : $\langle w, x \rangle + b = 1$ et $\langle w, x \rangle + b = -1$

Ces deux hyperplans sont appelés hyperplans canoniques. Notons que la marge des hyperplans canoniques est $1/||w||$. Le vecteur w possède à présent une signification géométrique trés claire.

[1] La direction de *w* donne le sens positif

Sample output to test PDF Combine only

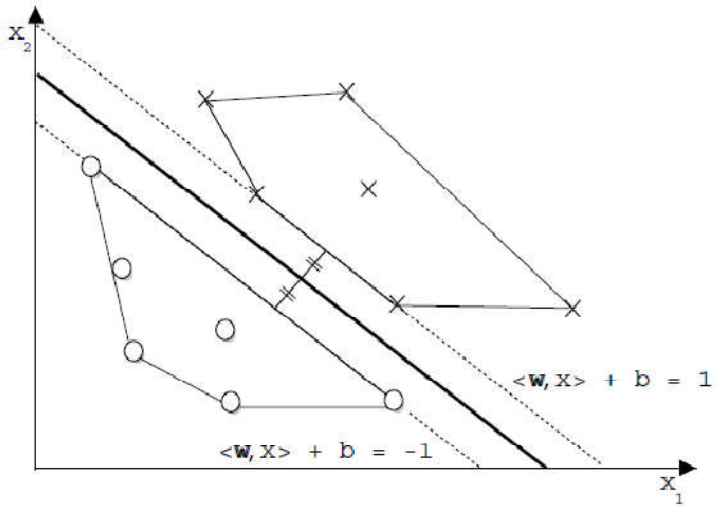


FIG.II.3 : Hyperplans Canoniques

II.4.4) Un problème d'optimisation

II.4.4.1) Le problème à résoudre par la SVM

II.4.4.1.1) Cas séparable

Supposant toujours que l'on dispose d'un *training set S* effectivement séparable par un séparateur linéaire. Si le séparateur sépare effectivement *S*, avec du côté positif les exemples étiquetés 1, et du côté négatif ceux étiquetés -1, alors toutes les marges des exemples sont positives (eq 3.b). Si l'une des marges est négative, c'est que le séparateur ne sépare pas correctement les deux classes, alors que c'est possible, vu qu'on suppose *S* linéairement séparable. Dans ce cas incorrect, la marge est négatif (eq 3.c). Donc maximiser la marge, veut bien dire d'abord qu'on sépare (marge positive), puis qu'on sépare bien (marge maximale).

Pour chaque séparateur, les exemples du *training set* les plus proches (eq 3.c) se trouvent sur les hyperplans canoniques de celui-ci, pour le séparateur à marge maximale ces exemples ont une marge plus grande que les exemples de plus petite marge des autres séparateurs possibles et sont appelés *vecteurs support*. En rappelle la distance qui sépare les vecteurs supports au plan de séparation, la marge donc, est $1/\|w\|$.

La largeur de la bande constituée par les hyperplans $\langle w,x\rangle + b = 1$ et $\langle w,x\rangle + b = -1$ est $2/\|w\|$. Pour trouver le séparateur de marge maximale, il suffit alors de chercher, parmi les séparateurs vérifiant pour tous les exemples $y*f(x) > 1$, le séparateur pour lequel $\|w\|$ est minimal.

Sample output to test PDF Combine only

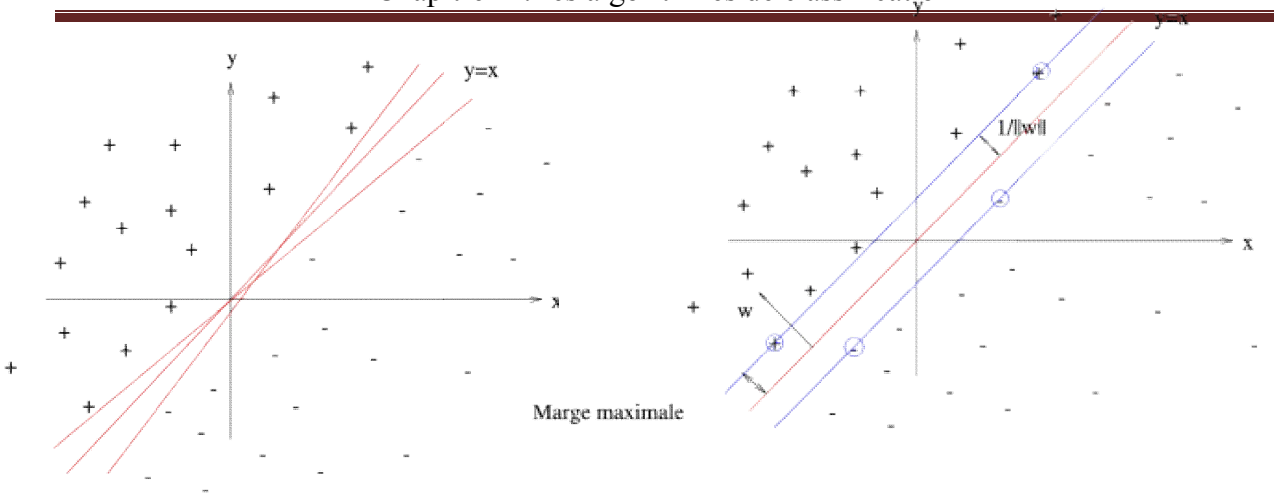


FIG.II.4 : Infinité d'hyperplans séparateurs, et l'hyperplan optimal avec marge maximale, Les échantillons entourés sont des vecteurs supports.

Maintenant, nous pouvons formuler un problème d'optimisation mathématique tel que sa solution nous fournisse l'hyperplan optimal (maximisant la marge) :

(QP1) Minimiser $W(w,b)=\frac{1}{2}\|x\|^2$

Tel que $y_i\left(\langle w,x_i\rangle+b\right)\geq 1$

Il s'agit d'un problème quadratique dont la fonction objective est à minimiser. Cette fonction objective est le carré de l'inverse de la double marge. L'unique contrainte stipule que les exemples doivent être bien classés et qu'ils ne dépassent pas les hyperplans canoniques.

Dans cette formulation, les variables à fixer sont les composantes *w* et *b*. D'un point de vue ML, ces variables correspondent aux α_i de la machine d'apprentissage. Le vecteur *w* possède un nombre de composantes égal à la dimension de l'espace d'entrée. En gardant cette formulation telle quelle, nous souffrons des même problèmes que les méthodes classiques du ML (*overfitting, curve of dimensionality*). Pour éviter cela, il est nécessaire d'introduire une formulation dite duale du problème. Un problème dual est un problème fournissant la même solution que le primal mais dont la formulation est différente. On appellera *variables primales*, les variables du problème primal, et *variables duales*, les variables du problème dual qui n'interviennent pas dans le primal.

Sample output to test PDF Combine only

Pour dualiser (QP1), nous devons former ce que l'on appelle le Lagrangien. Il s'agit de faire rentrer les contraintes dans la fonction objective et de pondérer chacune d'entre elles par une variable duale :

$$L(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{n}\alpha_i\left[y_i\left(\langle w,x_i\rangle+b\right)-1\right]$$

Les variables duales α_i intervenant dans le Lagrangien sont appelées *multiplicateurs de Lagrange*. Selon le type de contraintes qu'ils représentent, les multiplicateurs doivent respecter les règles de signes suivantes :

1- contrainte du type $C_i(x) \geq 0$: $\alpha_i \geq 0$
2- contrainte du type $C_i(x) = 0$: α_i est sans restriction de signe
3- contrainte du type $C_i(x) \leq 0$: $\alpha_i \leq 0$

En pratique, pour le dernier cas on considère des multiplicateurs positifs mais ces derniers interviennent avec un signe "–" dans le Lagrangien. On peut donner une signification physique à ces multiplicateurs : la variables α_i représente la "force" avec laquelle la solution appuie sur la contrainte i. Ainsi, un hyperplan qui violerait la contrainte pour x_i (il classe cet exemple du mauvais côté) rendrait α_i très grand ce qui ferait fortement augmenter la fonction objective (L). Cette solution ne pourrait donc pas être retenue comme solution optimale. Notons que L doit être minimisé par rapport aux variables primales et maximisé par rapport aux variables duals[2].

A présent, nous introduisons les conditions *Karush Kuhn et Tucker* (KKT) statuant sur l'optimalité d'une solution.

Théorème (KKT pour les problèmes différentiables convexes [Karush,1939], [Kuhn et Tucker,1951])

Considérons un problème d'optimisation de la forme :

Minimiser $g(x)$ tel que $\begin{cases} c_i(x) \leq 0 \forall i = 1..n \\ e_j(x) = 0 \forall j = 1..n' \end{cases}$

[2] En maximisant par rapport aux α_i , on assure qu'un maximum de contraintes soient satisfaites

Sample output to test PDF Combine only

Avec *g(.), $c_i(x)$ et $e_j(x)$* convexes et différentiables.

Le lagrangien est formé comme suit :

$$L(x,\alpha) = g(x) - \sum_{i=1}^{n} \alpha_i c_i(x) + \sum_{j=1}^{n'} \beta_j e_j(x)$$

La solution $\bar{x}$ est optimale SSI il existe $\bar{\alpha} \in R^n$ avec $\alpha_i \geq 0 \forall i = 1..n$ et $\bar{\beta} \in R^{n'}$ avec β_j s.r.s $\forall j = 1..n'$ tels que :

$$\partial_x L(\bar{x},\bar{\alpha}) = \partial_x g(\bar{x}) - \sum_{i=1}^{n} \alpha_i \partial_x c_i(\bar{x}) + \sum_{j=1}^{n'} \beta_j \partial_x e_j(\bar{x}) = 0$$

$$\partial_{\alpha_i} L(\bar{x},\bar{\alpha}) = c_i(\bar{x}) \leq 0$$

$$\partial_{\beta_j} L(\bar{x},\bar{\alpha}) = e_j(\bar{x}) = 0$$

$$\bar{\alpha}_i c_i(\bar{x}) = 0 \qquad \forall i = 1...n$$

$$\bar{\beta}_j e_j(\bar{x}) = 0 \qquad \forall j = 1...n'$$

Ce théorème fondamental en optimisation mathématique, nous fournit une condition suffisante et nécessaire pour l'optimalité d'une solution dans le cadre de problèmes différentiables convexes (ce qui est notre cas). Les deux dernières conditions sont souvent appelées *conditions KKT complémentaires*. Ces conditions expriment deux choses. Pour le voir prenons la première, $\bar{\alpha}_i c_i(\bar{x}) = 0$:

1. $\bar{\alpha}_i = 0$, Dans ce cas la solution n'est pas "sur la contrainte". Il n'y a donc rien à imposer au niveau de la solution.

2. $\bar{\alpha}_i \neq 0$, La solution est "sur la contrainte". Dans ce cas, la nullité du produit impose que la solution ne dépasse pas la contrainte (elle reste faisable).

Déterminons à présent les conditions KKT de notre problème d'optimisation (QP1).

Sample output to test PDF Combine only

$$\partial_w L(w,b,\alpha) = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \qquad (4.1)$$

$$\partial_b L(w,b,\alpha) = \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad (4.2)$$

$$\partial_{\alpha_i} L(w,b,\alpha) = -y_i\left(\langle w,x\rangle + b\right) + 1 \le 0 \qquad (4.3)$$

$$\alpha_i\left(y_i\left(\langle w,x\rangle + b\right) - 1\right) = 0 \qquad \forall i = 1..n \qquad (4.4)$$

L'équation (4.1) permet de réexprimer *w* :

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \qquad (4.5)$$

Remarquons qu'avec cette formulation, on peut calculer *w* en fixant seulement *n* paramètres. **L**'idée va donc être de formuler un problème dual dans lequel *w* est remplacé par sa formulation

$$L(w,b,\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle$$

(4.5). **D**e cette façon, le nombre de paramètres à fixer est relatif au nombre d'exemples du *training set* et non plus à la dimension de l'espace d'entrée (supposée très élevée). Pour ce faire, nous substituons (4.2) et (4.5) dans le Langrangien :

A partir de quoi nous pouvons formuler le problème dual :

$$(QP2) \qquad \textit{Maximiser} \quad W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle$$

$$\textit{tel que} \quad \begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0 \\ \alpha_i \ge 0 \qquad \forall i = 1...n \end{cases}$$

La résolution du dual permet donc de calculer le vecteur *w* à moindre coût, cependant cette formulation ne fait à aucun moment apparaître le terme *b*. Pour calculer ce dernier nous devons utiliser les variables primales :

Sample output to test PDF Combine only

$$b = -\frac{\max_{y=-1}\left(\langle w, x_i \rangle\right) + \max_{y=+1}\left(\langle w, x_i \rangle\right)}{2}$$

Nous avons à présent tous les éléments nécessaires pour exprimer la fonction de décision de notre classificateur linéaire :

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \langle x, x_i \rangle + b$$

Notons qu'un grand nombre de termes de cette somme sont nuls. En effet seuls les α_i correspondant aux exemples se trouvant sur les hyperplans canoniques ("sur la contrainte") sont non-nuls. Ces exemples sont appelés *Support Vectors* (SV). On peut les voir comme les représentants de leurs catégories car si le *training set* n'était constitué que des SV, l'hyperplan optimal que l'on trouverait serait identique.

II.4.4.1.2) Marge souple

En général, il n'est pas non plus possible de trouver une séparatrice linéaire dans l'espace de redescription. Il se peut aussi que des échantillons soient mal étiquetés, et que l'hyperplan séparateur ne soit pas la meilleure solution au problème de classement.

[Cortes et Vapnik, 1995] proposent une technique dite de marge souple, qui tolère les mauvais classements. La technique cherche un hyperplan séparateur qui minimise le nombre d'erreurs grâce à l'introduction de *variables ressort* ξ_k (*slack variables en anglais*), qui permettent de relâcher les contraintes sur les vecteurs d'apprentissage:

Sample output to test PDF Combine only

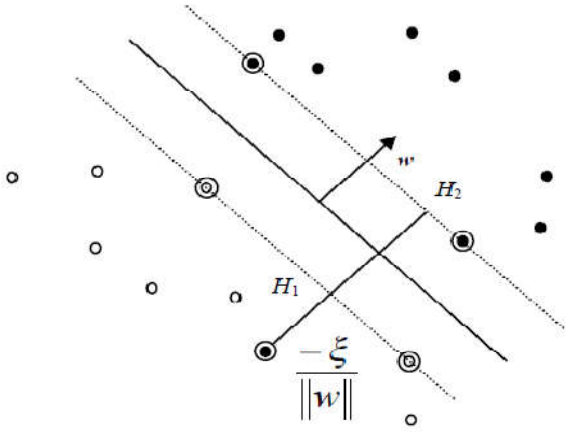


FIG.II.5 : Les hyperplans linéaires pour un problème de classification non linéairement séparable. Lorsqu'il y a une erreur sur un exemple, cet exemple est considéré comme un vecteur support dont sa distance avec l'hyperplan de sa vraie classe est -□/||w||.

$$y_i\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i \qquad (4.6)$$

$$\forall i, \quad \xi_i \geq 0 \qquad (4.7)$$

D'après (4.6) et (4.7) lorsqu'il y a une erreur pour un point x_i donné, la variable ξ_i correspondante doit être plus grande que l'unité, puisque dans ce cas +1-ξ_i ou -1+ξ_i changent de signe. Par ce fait $\sum_i \xi_i$ détermine une borne supérieure du nombre d'erreurs en apprentissage. Ainsi, une façon simple de traiter ce cas est de changer la fonction objective du problème primal de $\frac{\|w\|^2}{2}$ en $\frac{\|w\|^2}{2} + C$, où C est un paramètre fixé à l'avance. Plus C est grand plus on pénalise les erreurs. La figure (4.2.1) illustre ce cas.

Comme précédemment le lagrangien de ce problème s'écrit :

$$(QP3) \qquad \textit{Minimiser} \qquad L(w,b,x_i) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\textit{tel que} \quad \begin{cases} y_i\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i & \forall i = 1..n \\ \xi_i \geq 0 & \forall i = 1..n \end{cases}$$

Sample output to test PDF Combine only

En formant le Lagrangien, puis en appliquant le théorème (KKT), on trouve la forme duale suivante :

$$(QP4) \qquad Maximiser \quad W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle$$

$$tel\ que \quad \begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0 \\ 0 \le \alpha_i \le C \qquad \forall i = 1...n \end{cases}$$

Nous terminons en mentionnant que l'on peut à nouveau déterminer le statut d'un exemple x_i en regardant sa variable duale α_i. Cette fois-ci il existe trois statuts différents :

1. $\alpha_i = 0$: L'exemple est bien classé et n'est pas sur un des deux hyperplans canoniques. On dira que l'exemple est un non-SV.

2. $0 < \alpha_i < C$: L'exemple est bien classé et se trouve sur un hyperplan canonique. Il s'agit donc d'un SV.

1. $\alpha_i = C$: L'exemple est mal classé. Il sera malgré tout considéré comme SV puisque $\alpha_i > 0$. Il s'agit d'un *outlier*.

II.4.5) Les fonctions Noyaux (Kernels)

Le gros intérêt des noyaux est que tout ce qu'on vient de voir sur la séparation linéaire s'applique en fait très facilement à des séparations non linéaires, sous réserve de bien faire les choses.

II.4.5.1) L'espace des caractéristiques

Imaginons un ensemble d'exemples x_i étiquetés par -1 ou +1 suivant la classe à laquelle ils appartiennent, qui ne soient pas du tout séparable linéairement. La méthode vue au section précédente fonctionne, mais la séparation est bien entendu de piètre qualité, et bon nombre de vecteurs sont des supports.

Sample output to test PDF Combine only

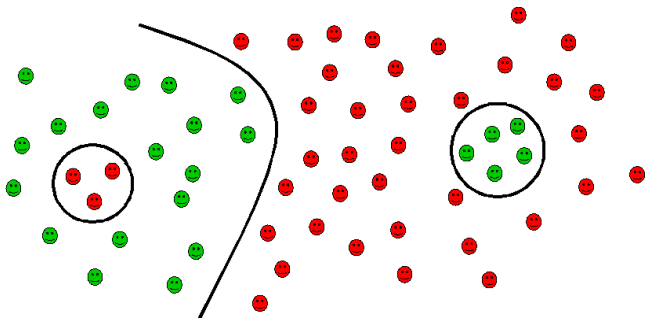


FIG.II.6 : Exemples non séparables linéairement

Une solution pour mieux séparer les exemples est de les projeter dans un espace différent (le *feature space*)[3], et de réaliser une séparation linéaire dans cet espace-là, où cette fois-ci elle devrait être plus adaptée.

Nous noterons le *feature space F*, et le *mapping* Φ vers cet espace, on a : $\Phi : X \to F$

$$\Phi(x) = \begin{pmatrix} \varnothing_1(x) \\ \varnothing_2(x) \\ . \\ . \\ \varnothing_n(x) \end{pmatrix}$$

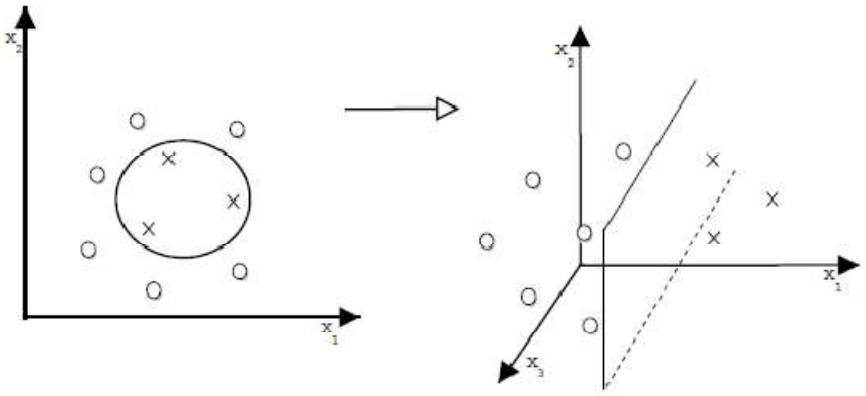


FIG.II.7 : Un mapping Φ rendant les exemples linéairement séparables

[3] Souvent de dimension très élevée

Sample output to test PDF Combine only

Les fonctions $\varnothing_i$ ne sont pas nécessairement linéaires, et on peut même avoir n = ∞ ! Si l'on reprend les méthodes de la section précédente, et que l'on travaille dans l'espace des caractéristiques (*feature space*), c'est-à-dire si on travaille avec le corpus

$$\overline{S} = \{(\Phi(x_i), y_i)\}_{0\le i\le n} \quad avec \quad \forall i = 1..n \quad y_i \in \{-1,1\}$$

Au lieu de $S = \{(x_i, y_i)\}_{0\le i\le n}$ *avec* $\forall i = 1..n$ $y_i \in \{-1,1\}$

On se retrouve à faire de la séparation linéaire sur le corpus $\overline{S}$. On obtient un séparateur, donné par la formule $w = \sum_{i=1}^{n} \alpha_i y_i x_i$ et le *b*. Pour décider de la classe d'un vecteur *x*, on pourrait calculer alors $\Phi(x)$, que l'on passerait à son tour comme argument du séparateur pour en connaître la classe +1 où −1.

En fait, on ne procédera pas comme ça, et on préférera éviter le calcul explicite de $\Phi(x)$ en remarquant que le problème d'optimisation posé à la section précédente ne fait intervenir les vecteurs que via des produits scalaires entre eux.

Notons *k(x,z)* le produit $\langle \Phi(x), \Phi(z) \rangle$. Travailler sur le corpus $\overline{S}$ revient à travailler sur le corpus S avec les algorithmes précédents, mais en remplaçant toutes les occurrences de $\langle \bullet, \bullet \rangle$ par $k(\bullet, \bullet)$.

Pour l'instant, on ne voit pas trop l'intérêt, vu que pour calculer *k(x,z),* il faut appliquer la définition, à savoir projeter *x* et *z* dans l'espace des caractéristiques et calculer le produit scalaire, dans cet espace, des deux vecteurs obtenus.

La ruse en fait est qu'on ne fera pas cette projection, car on calculera *k (x, z)* autrement. En fait, *k (x, z)* est une fonction que l'on va se donner, en s'assurant qu'il existe bien en théorie une projection Φ dans un espace qu'on ne cherchera pas à décrire. Ainsi, on calculera directement *k (x, z)*, à chaque fois que l'algorithme précédent requiert un produit scalaire, et c'est tout ! La projection dans le gros espace de caractéristiques sera implicite.

Sample output to test PDF Combine only

II.4.5.2) Conditions pour avoir un noyau

Il y a des conditions mathématiques, appelées théorème de Mercer, qui permettent de dire si une fonction est un noyau ou non, sans construire la projection dans l'espace des caractéristiques.

Théorème de Mercer : la fonction $k(x,z) : X \text{ x } X \longrightarrow R$ est un noyau SSi :

$$G = \left(k\left(x_i, x_j\right)\right)_{i,j=1}^{n} \quad est \quad définie \quad positive$$

Notons qu'une fonction $X \text{ x } X \longrightarrow R$ générant une matrice définie positive possède les trois propriétés fondamentales du produit scalaire : $\forall x_i, x_j \in X$

1. poitivité : $k\left(x_i, x_j\right) \geq 0$
2. symétrie : $k\left(x_i, x_j\right) = k\left(x_j, x_i\right)$
3. inégalité de Cauchy-Shwartz : $\left|k\left(x_i, x_j\right)\right| \leq \|x_i\| . \|x_j\|$

La condition de Mercer nous indique si une fonction est un *kernel* mais nous n'avons aucun renseignement sur le mapping Φ (et donc sur le *feature space*) induit par ce *kernel*. A la section suivante, nous présentons deux *kernels* considérés comme standards. Nous verrons que la nature du *feature space* est un peu particulière.

II.4.5.3) Exemples de kernels

II.4.5 .3.1) Kernel polynomial

La forme générique de ce kernel est : $k(x,z) = \left(a * \langle x, z \rangle + b\right)^d$

et correspond à une projection $\Phi(x)$ dans un espace de caractéristiques où chaque composantes $\Phi_i(x)$ est un monôme de degré inférieur à d de certaines des composantes de x. **L**e séparateur calculé avec ce noyau est un polynôme de degré d dont les monômes sont les composantes de x. La constante c, quand elle est élevée, donne plus d'importance aux monômes de degré élevé.

Sample output to test PDF Combine only

II.4.5.3.2) Le kernel RBF (Radial Basis Function)

La forme générique de ce kernel est : $k(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$

L'espace d'arrivée *F* (le *feature space*) de la fonction Φ est de dimension infinie étant donné que Φ fait correspondre une fonction continue à chaque exemple, Le *kernel RBF* permet donc de calculer des similarités dans un espace de dimension infinie.

Avec le kernel *RBF*, tous les exemples sont placés sur une sphère de rayon 1 :

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

Le paramètre σ permet de régler la largeur de la gaussienne. En prenant un σ grand, la similarité d'un exemple par rapport à ceux qui l'entoure sera assez élevée, alors qu'en prenant un σ tendant vers 0, l'exemple ne sera similaire à aucun autre. En resserrant fortement la gaussienne, un classificateur (faisant usage de ce kernel) peut arriver à apprendre n'importe quel *training set* sans commettre d'erreur. On sent tout de suite que le danger de l'apprentissage par cœur n'est pas loin. En fait, σ est un autre paramètre permettant de contrôler la capacité d'un classificateur.

II.4.5.3.3) Composition des kernels

Il est possible de composer des nouveaux kernels en utilisant des kernels existants. En prenant $k_1(.,.)$ et $k_2(.,.)$ des fonctions satisfaisant à la condition de Mercer, $a \in R^+$ et B une matrice définie positive, alors les fonctions suivantes sont des kernels :

1. $k(x,z) = k_1(x,z) + k_2(x,z)$
2. $k(x,z) = ak_1(x,z)$
3. $k(x,z) = k_1(x,z)k_2(x,z)$
4. $k(x,z) = x^T Bz$

Sample output to test PDF Combine only

II.4.6) Formulation de SVM

II.4.6.1) Cas linéairement séparable

La méthode SVM consiste en un classificateur à marge maximale dans lequel le produit scalaire a été remplacé par le kernel. Effectuons dès lors cette substitution dans le problème (QP2).

$$(QP5) \qquad Maximiser \quad W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j k(x_i, x_j)$$

$$tel\ que \quad \begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0 \\ \alpha_i \geq 0 \qquad \forall i = 1...n \end{cases}$$

II.4.6.2) Formulation Soft Margin

Cela revient à relaxer la contrainte imposant que tous les exemples soient bien classés.

$$(QP6) \qquad Maximiser \quad W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j k(x_i, x_j)$$

$$tel\ que \quad \begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \qquad \forall i = 1...n \end{cases}$$

Conclusion

Les techniques d'apprentissage artificiel supervisé ont été appliquées dans plusieurs domaines. Cependant, le choix de la meilleure technique dépend de la tâche en cours. Pour pouvoir répondre à cette question le même chercheur doit comparer la fidélité des différentes techniques

Puis, nous avons détaillé la méthode principale de ce mémoire : SVM. Tout d'abord nous avons mis en évidence la relation qui existe entre la marge (de l'hyperplan séparateur) des classificateurs linéaires et la théorie de l'apprentissage statistique. Plus spécialement, nous avons vu que le fait de maximiser la marge permettait de réduire la capacité des fonctions du classificateur. Nous avons ensuite formulé la recherche de l'hyperplan optimal sous forme

Sample output to test PDF Combine only

d'un problème d'optimisation quadratique. Il s'est avéré que l'on pouvait en donner une représentation duale avantageuse. Principalement, cette formulation permet de limiter le nombre d'inconnus du problème au nombre d'exemples du *training set*. Cela constitue une des avantages de la méthode SVM par rapport aux autres méthodes face à la course aux dimensions. La solution apportée par SVM pour traiter des problèmes non-linéairement séparables consiste à utiliser les fonctions *kernels*. L'avantage principal de ces fonctions est de pouvoir apprendre un modèle de classification pour des exemples non linéairement séparables.

Sample output to test PDF Combine only

Introduction

Les Machines à Vecteurs de Support (SVM) sont une méthode très populaire d'apprentissage supervisé pour la classification. La façon standard pour résoudre le problème d'optimisation avec contraintes des SVMs consiste à introduire des multiplicateurs de Lagrange pour chaque contrainte, et d'optimiser le problème dual équivalent qui est une instance de programmation quadratique. L'optimisation directe de ce problème quadratique devient rapidement coûteux lorsque le nombre d'exemples dans la base d'apprentissage augmente. Pour remédier à ce problème, des études ont porté sur l'optimisation efficace de la forme duale par décomposition du problème d'apprentissage **(Osuna et al., 1997)**. Les méthodes à base de décomposition comme SMO **(Platt, 1998)**, SVM-light **(Joachims, 1999)**.

III.1) Les méthodes de décomposition

Reprenant la version matricielle du problème d'optimisation (QP6) :

$$(QP) \qquad Minimiser \quad W(\alpha) = -\alpha^T 1 + \frac{1}{2}\alpha^T Q \alpha \qquad (6.4)$$

$$tel\ que \quad \begin{cases} \alpha^T y = 0 & (6.5) \\ 0 \le \alpha \le C1 \quad \forall i = 1...n & (6.6) \end{cases}$$

Où la matrice Q est défini par $Q_{ij} = y_i y_j k(x_i, x_j)$

La taille du problème d'optimisation dépend du nombre d'exemples du training set noté l. La taille de la matrice Q est l^2, pour un *set training* de 10000 exemples et plus, il devient impossible de maintenir Q en mémoire. Plusieurs implémentations des solutions algorithmiques pour (QP) nécessitent un stockage explicite de Q ce qui empêche leur application.

III.2) Algorithmes de décomposition

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

Sample output to test PDF Combine only

La propriété utilisée : la parcimonie

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

$$\alpha_i = 0 \quad \rightarrow \quad y_i f(x_i) > 1$$

$$0 < \alpha_i < C \quad \rightarrow \quad y_i f(x_i) = 1$$

$$\alpha_i = C \quad \rightarrow \quad y_i f(x_i) < 1$$

Le principe général des méthodes de décomposition et/ou de contraintes actives repose sur l'observation que seuls les points non contraints dans la solution nécessitent le calcul de leur coefficient : c'est la parcimonie. En effet, les autres ont pour coefficient une valeur fixe donnée par le problème : celle de la borne qui les contraint. Ce constat a amené à différentes techniques de décompositions et différents algorithmes que nous allons détailler.

III.2.1) L'algorithme SMO (Sequential Minimal Optimization)

L'algorithme *Sequential Minimal Optimization* (SMO) posé par **[Platt, 1998]** est un algorithme qui permet de résoudre rapidement le problème quadratique (QP4) du SVM sans passer par toutes les étapes de résolution numérique d'un QP. L'idée principale des algorithmes de décomposition est de travailler avec un sous-ensemble réduit de données du problème, garder les solutions et continuer avec le reste des données où les solutions antérieures doivent être encore testées. La SMO prend cette idée à l'extrême : elle optimise seulement deux vecteurs par itération. Cette optimisation admet une solution analytique. A chaque itération, la SMO choisit deux coefficients de Lagrange α_i et α_j pour les optimiser ensemble, trouver ses valeurs optimales étant donné que toutes les autres sont fixes, et actualiser le vecteur solution α.

III.2.2) L'algorithme de décomposition D'Osuna

L'idée générale de la technique de décomposition de **[Osuna et al. 1997a]** est de garder une taille de sous ensemble constante en laissant la possibilité de retirer du sous ensemble

Sample output to test PDF Combine only

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

Les conditions d'optimalité

Le problème quadratique que nous devons résoudre est le suivant :

$$(QP4) \qquad Minimiser \quad W(\alpha) = -\alpha^T 1 + \frac{1}{2}\alpha^T Q\alpha$$

$$tel\ que \quad \begin{cases} \alpha^T y = 0 \\ -\alpha \le 0 \quad \forall i = 1...n \\ \alpha - C \le 0 \end{cases}$$

Donc les conditions de Kuhn-Tuker sont nécessaires et suffisantes pour l'optimalité. Les conditions de KT sont comme suit :

$$\nabla W(\alpha) + \left(\lambda^{eq} y - \lambda^{lo} + \lambda^{up}\right) = 0$$
$$\forall i = 1..n \qquad \lambda_i^{lo}\left(-\alpha_i\right) = 0$$
$$\forall i = 1..n \qquad \lambda_i^{up}\left(\alpha_i - C\right) = 0$$
$$\alpha^T y = 0$$
$$\lambda^{up} \ge 0 \qquad (1)$$
$$\lambda^{lo} \ge 0 \qquad (2)$$
$$0 \le \alpha \le C1$$

Afin de dériver encore d'autres expressions algébriques des conditions d'optimalité (1) et (2) nous assumons l'existence de certains α_i tels que $0 < \alpha_i < C$, et considérer les trois valeurs possibles que chaque composant de α_i peut avoir :

<u>Cas 1 :</u> $0 < \alpha_i < C$ pour les trois premières équations des conditions de KT nous avons :

$$(Q\alpha)_i - 1 + \lambda^{eq} y_i = 0 \qquad (3)$$

En utilisant les résultats de **[Cortes et al, 1995]** et **[Vapnik, 1995],** on peut facilement prouver que quand α est strictement entre 0 et C l'égalité suivante:

Sample output to test PDF Combine only

$$y_i\left(\sum_{j=1}^{l}\alpha_j y_j k(x_i,x_j)+b\right)=1 \qquad (4)$$

Et on a $(Q\alpha)_i = y_i\left(\sum_{j=1}^{l}\alpha_j y_j k(x_i,x_j)\right)$

En combinant cette expression avec (3) et (4) nous obtenons immédiatement $\lambda^{eq} = b$.

Cas 2 : $\alpha_i = C$

Par la définition $f(x_i)=\left(\sum_{j=1}^{l}\alpha_j y_j k(x_i,x_j)+b\right)$

et notant cela $(Q\alpha)_i = y_i\left(\sum_{j=1}^{l}\alpha_j y_j k(x_i,x_j)\right) = y_i(f(x_i)-b)$

nous concluons que $y_i f(x_i)\le 1$

Cas 3 : $\alpha_i = 0$

En appliquant une manipulation algébrique semblable en tant que celle décrite pour le cas 2, nous obtenons $y_i f(x_i)\ge 1$ (5)

Stratégie de décomposition

Les conditions d'optimalité dérivés dans la section précédente sont essentielles afin de concevoir une stratégie de décomposition qui prend l'avantage que la majorité des α_i sont nuls et ce garantit qu'à chaque itération la fonction objective prend pas vers le minimum. Afin d'accomplir ce but on divise l'ensemble des variables en deux ensembles B et N de telle manière que les conditions d'optimalité se tiennent dans le sous-problème défini seulement pour les variables dans l'ensemble B, qu'on appele *working set,* puis on décompose le vecteur α en deux vecteurs α_B et α_N et mettre $\alpha_N = 0$. En utilisant cette décomposition les rapports suivants sont clairement vrais :

Sample output to test PDF Combine only

- **O**n peut remplacer $\alpha_i = 0 \quad i \in B$, avec $\alpha_j = 0 \quad j \in N$, sans changer le coût de la fonction ou la praticabilité du sous-problème et du problème original.
- **A**près un tel remplacement, le nouveau sous-problème est optimal si et seulement si $y_i f(x_i) \geq 1$ Ceci résulte de l'équation (5) et de la prétention que le sous-problème était optimal avant que le remplacement ait été fait.

Les rapports précédents mènent à la proposition suivante :

Proposition :

Etant donné une solution optimale d'un sous-problème défini sur B, l'opération du remplacement $\alpha_i = 0 \quad i \in B$ avec $\alpha_j = 0 \quad j \in N$ vérifiant $y_i f(x_i) < 1$ génère un nouveau sous-problème qui une fois optimisé, rapporte une amélioration stricte de la fonction objective. Une preuve détaillée de cette proposition peut être trouvée dans **[Osuna et al. 1997b]**.

L'algorithme de décomposition

Supposons que nous pouvons définir un *working set* B à taille fixe, tel que $|B| \leq l$, et il est assez grand pour contenir tous les vecteurs de support, mais assez petit tels que la machine peut le manipuler et l'optimiser en utilisant un certain QP solveur. Alors l'algorithme de décomposition peut être énoncé comme suit :

1. Choisir arbitrairement $|B| \leq l$ points de l'ensemble d'apprentissage.
2. Résoudre le sous-problème défini par les variables dans le B.
3. tant qu'il existe $j \in N$, tel que $y_i f(x_i) < 1$, remplacer $\alpha_i = 0 \quad i \in B$ avec $\alpha_j = 0 \quad j \in N$ et résoudre le nouveau sous-problème.

Notons que, selon les conditions d'optimalité décrites ci-dessus, cet algorithme amélioreront strictement la fonction objective à chaque itération et donc ne feront pas un cycle. Puisque la fonction objective est limitée l'algorithme doit converger à la solution optimale globale dans un nombre fini d'itérations. **FIG.II.5.5.1.1** donne une interprétation géométrique de la manière que l'algorithme de décomposition permet la redéfinition de la surface de séparation en ajoutant les points qui violent les conditions d'optimalité.

Sample output to test PDF Combine only

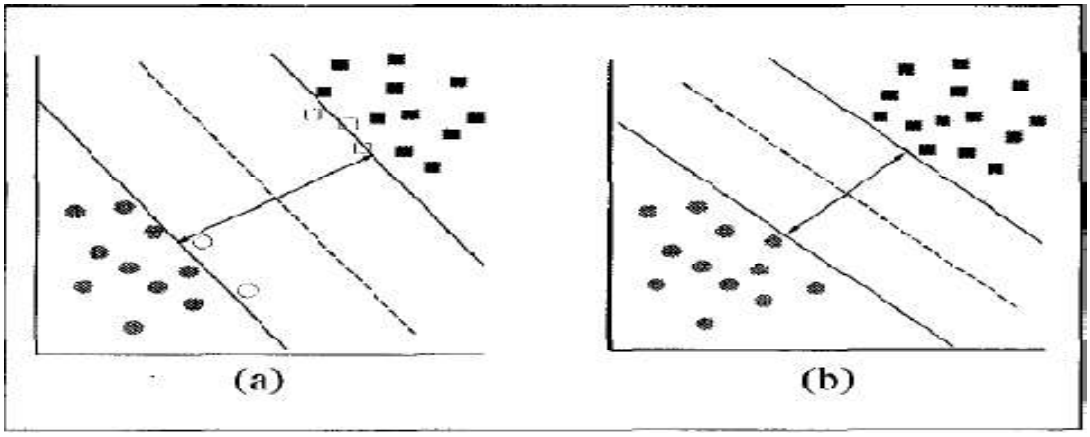


FIG.III.1 : (a) Une solution sous-optimale où les points non-remplis ont $\alpha = 0$ mais violent des conditions d'optimalité en étant à l'intérieur de l'intervalle [-1,+1]. (b) La surface de décision est redéfinie. Depuis aucuns points avec $\alpha = 0$ sont à l'intérieur de l'intervalle [-1,+1], la solution est optimale. Noter que la taille de la marge a diminué, et la forme de la surface de décision a changé.

III.2.3) L'algorithme SVMlight

Le SVM^{light} est un algorithme proposé par **[Joachims, 1998]**, qui utilise l'idée de décomposition de **[Osuna et al. 1997a]**. L'avantage principal de cette décomposition est qu'il suggère des algorithmes avec des conditions de mémoire linéaires dans le nombre d'exemples d'apprentissage et linéaires dans le nombre de vecteurs supports. Un inconvénient potentiel est que ces algorithmes peuvent avoir besoin d'un long temps d'apprentissage. Pour aborder ce problème, **[Joachims, 1999]** propose un algorithme qui incorpore les idées suivantes :

- Une méthode efficace pour choisir le *working set*.
- « Shrinking » successif du problème d'optimisation. Ceci exploite la propriété que
  - beaucoup moins de vecteurs supports (SVs) que des exemples d'apprentissage.
  - Beaucoup de SVs qui ont un α_i à la limite supérieure C.
- L'amélioration de techniques de calcul tel que cacher et mettre à jour d'une façon incrémentale le gradient et les critères d'arrêt.

Shrinking Le shrinking **[Joachims, 1999]** est une heuristique visant à déterminer au cours de l'avancement de l'algorithme quels points seront certainement exclus de la solution ou bornés. On connaîtra leur valeur sans avoir à calculer leur coefficient α_i. De cette façon, on peut ne plus tenir compte de ces points et réduire mécaniquement la taille du problème à résoudre.

Sample output to test PDF Combine only

Comme cette heuristique est faillible, il faut vérifier à l'arrêt de l'algorithme que les points exclus sont dans le bon groupe I_0 ou I_C et éventuellement de refaire une étape d'optimisation.

noisulcnoC

Dans ce chapitre, nous avons introduit des approches reposent sur l'idée de décomposition pour résoudre le problème d'optimisation quadratique avec des contraintes d'inégalité et une contrainte linéaire d'égalité. Cette décomposition dédouble (*QP*) en deux, une partie inactive et une autre active que nous appellerons par la suite le *working set*. L'avantage principal de cette décomposition est qu'il suggère des algorithmes avec des conditions de mémoire linéaires dans le nombre de SV_S.

Sample output to test PDF Combine only

IV.1) Introduction

Dans les chapitres précédents, on a illustré en détails les différentes techniques de représentation, ainsi que les approches théoriques utilisées dans le cadre de la conception de notre programme de classification de document par SVM.

Dans ce qui suit, On va lister tout d'abord les sources de corpus et sa description. Puis, on va voir les méthodes sur lesquelles on s'est basé pour exploiter ces différents concepts mathématiques afin de réaliser un programme de classification de documents robuste et fiable, ainsi qu'une présentation détaillée des différents modules qui le composent et la manière dont on a implémenté les algorithmes

IV.2) Évaluation des résultats du classifieur

Pour évaluer les résultats obtenus pas un classifieur, les documents de l'espace d'apprentissage sont souvent divisés en deux ensembles : le premier est utilisé pour la construction du classifieur tandis que le deuxième est utilisé pour faire le test. Puisqu'on adopte l'approche de classification supervisée on connait à l'avance la catégorie de chaque document. Ainsi, on compare la catégorie prédite avec celle prédéfinie et on calcul un score de performance. Ce calcul peut se faire de diverses façons.

IV.3) Analyse des besoin

Notre tendance est de développer un logicielle capable de classer des documents dans les domaines d'application typique sont : La classification automatique des documents, La recherche d'information etc…., d'où ce logiciel est implémenté par SVM (Supports Vectors Machine)

Dans ce chapitre, on tient à détailler la partie pratique de notre travail, d'où on va présenter au début un schéma global de l'application, puis on essayera de détailler chaque partie séparément.

Sample output to test PDF Combine only

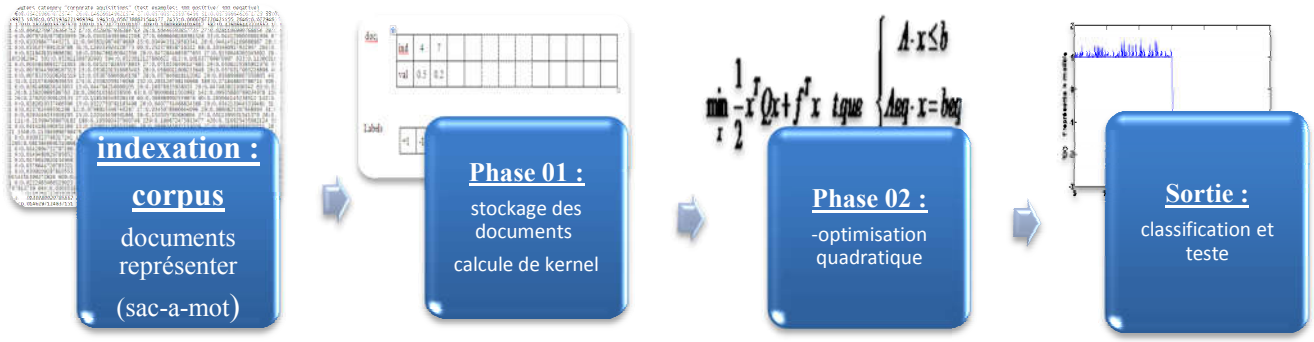


FIG IV.1: schéma d'application

IV.3.1) l'indexation des documents

Notre algorithme est apprêté à achever de la classification de documents textuels. Notre sélection de représentation s'est certainement porté sur le modèle vectoriel. Dans le cas de la représentation *bag of words*

L'ensemble d'apprentissage pratiqué est un modèle de la base de donnée "Reuters-21578" disponible sur http://download.joachims.org/svm_light/examples/example1.tar.gz, cette unité contient des exemples positifs (classe +1) et exemples négatifs (classe -1) .

Corpus d'apprentissage

CORPUS	CATEGORIES	#DE DOCUMENTS	TOTALE
train20	+1	10	20
	-1	10	
Train100	+1	50	100
	-1	50	
Train600	+1	300	600
	-1	300	
Train800	+1	400	800
	-1	400	
teste	+1	300	600
	+1	300	

Sample output to test PDF Combine only

Tab IV. 1 Corpus d'apprentissage

IV.3.2) Stockage des documents et calcul des kernels

Dans le cas de la représentation *bag of words*, les vecteurs exposent un nombre très élevé de composantes nulles. Cette épreuve nous pousse à dévisager une manière de stocker les exemples de facon avantageux .favorable (FIG **IV.2**)

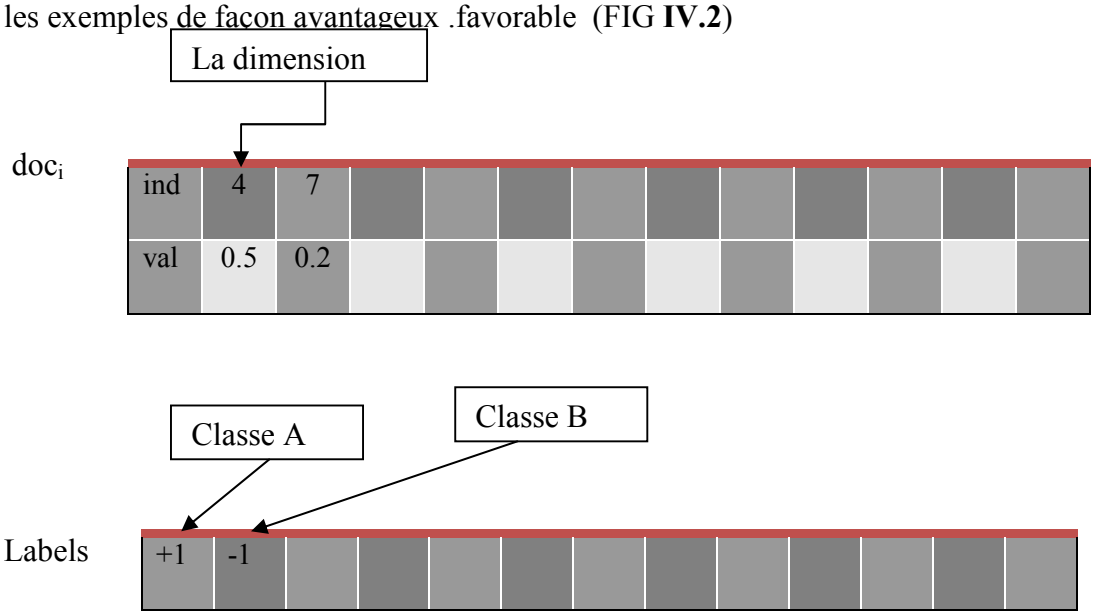


FIG IV.2: Structure de donnée utilisée pour représenter un document

IV.3.3) Optimisation quadratique

Dans les phases de prédiction, il est à chaque fois question de résoudre un système quadratique : le système réduit. Dans Matlab, l'optimiseur est la fonction *quadprog(),*qui cherche le minimum pour un problème spécifié par

$$\min_x \frac{1}{2} x^T Q x + f^T x \quad t.que \quad \begin{cases} A \cdot x \le b \\ Aeq \cdot x = beq \\ lb \le x \le ub \end{cases}$$

Q, A, et ***Aeq*** sont des matrices, et ***f, b, beq, lb, ub,*** et ***x*** sont des vecteurs.

Sample output to test PDF Combine only

x = quadprog (H,f,A,b,Aeq,beq)

Pour notre problème nous avons : $Q_{ij} = y_i y_j K(x_i, x_j)$, $f(i) = -1$, $Aeq(1,i) = y_i$, $beq(i) = 0$, $lb(i) = 0$ et $ub(i) = c$

IV.3.4) Implémentation de l'algorithme

Au début nous avons choisi arbitrairement $\frac{|B|}{2}$ éléments de la classe $\{+1\}$ et $\frac{|B|}{2}$ éléments de la classe $\{-1\}$. Il est facile de calculer les données du système quadratique sauf la hessienne Q qui est une matrice symétrique. La méthode proposée requiert que Q soit stockée sous forme d'une matrice triangulaire supérieure droite. Le nombre d'éléments occupés est donné par $\frac{n(n+1)}{2}$ où n est le nombre d'éléments qui ont visité l'ensemble B ($n \geq |B|$). Pour le kernel nous avons pris le *RBF* avec $\delta = 10$.

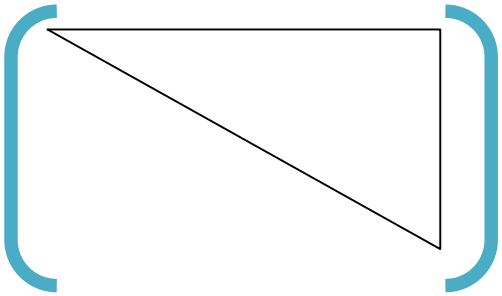


FIG IV.3: Représentation de la hessienne Q triangulaire

Pour faire les tests des éléments à remplacer et le test d'arrêt, nous avons utilisé les vecteurs X contenant les nouvelles valeurs des variables α_i comme solution de l'optimiseur *quadprog()*, FX pour calculer les valeurs $y_i f(x_i)$, et enfin le vecteur FXN contenant les valeurs $y_j f(x_j)$ $j \in N$ trié d'un ordre croissant pour chercher α_j $j \in N$ $y_j f(x_j) < y_k f(x_k)$ $\forall k \in N$.

Sample output to test PDF Combine only

IV.4)Implémentation

IV.4.1)Le langage de programmation :

Dans cette application nous avons utilisé le langage : MATLAB R2008a version7.6.0.324

MATLAB (Matrix Laboratory) est un langage de développement informatique particulièrement dédié aux applications scientifiques.

MATLAB est utilisé pour développer des solutions nécessitant une très grande puissance de calcul. il offre aujourd'hui bien d'autres possibilités. Il contient des bibliothèques spécialisées (toolbox) qui répondent à des besoins spécifiques. Sa facilité d'apprentissage et d'utilisation (due à une syntaxe très claire) en ont fait un standard adapté pour les divers problèmes d'ingénierie.

Parmi les raisons qui nous ont poussés à l'utiliser, on trouve :

• Ses très nombreuses fonctions prédéfinies et prêtes à l'emploi.

• Sa simplicité à l'implémentation et rapidité de calculs.

• Sa fiabilité et sa robustesse.

IV.4.2)Présentation du logiciel

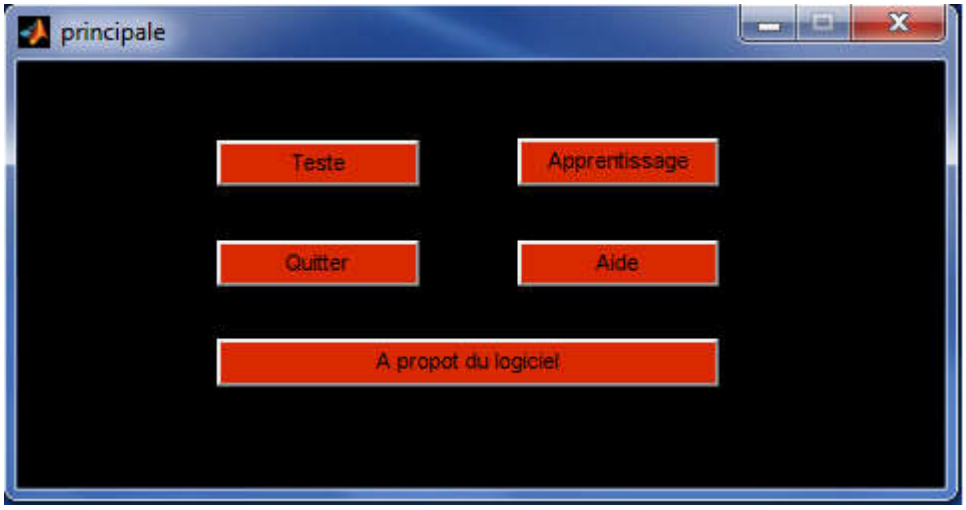


FIG IV.4: interface principale de logiciel

Sample output to test PDF Combine only

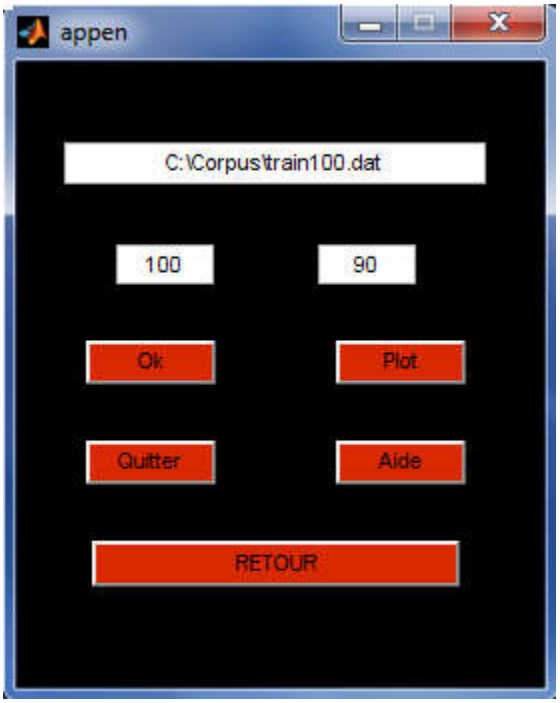


FIG IV.5*:* fenêtre d'apprentissage

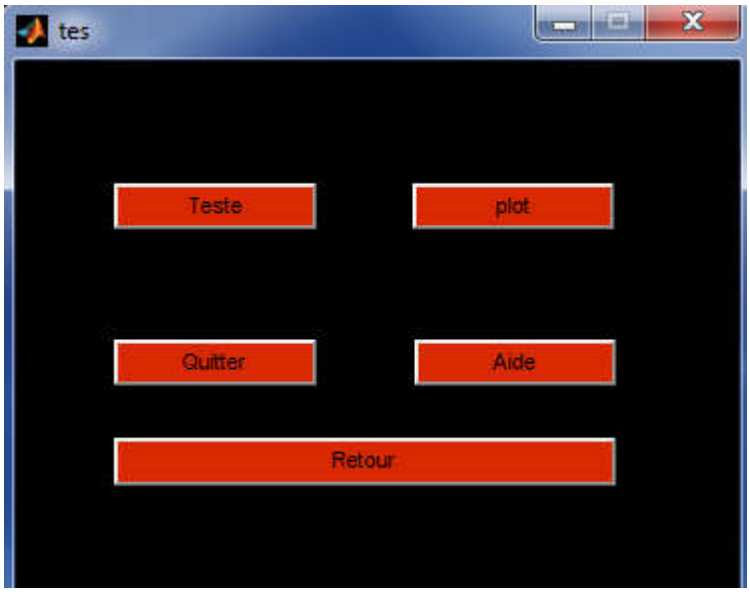


FIG IV.6*:* fenêtre de test

Sample output to test PDF Combine only

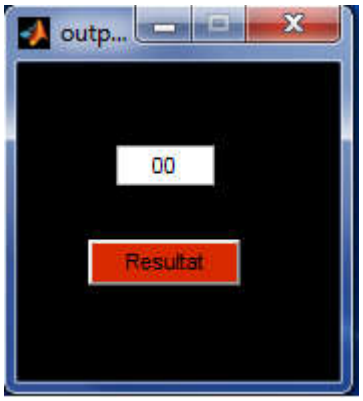


FIG IV.6*:* fenêtre de resultat

IV.5)Expériences

Les expériences sont menées de façon progressives vers selon plusieurs paramètres de SV et CORPUS.Tous les résultats sont argumentés et discuté d'une manière globale.

nombre de documents	nombre VS	taux de performance
100	68	70,66%
600	441	96,66%
800	484	97.16 %

Tab IV.2: expériences selon nombre de documents

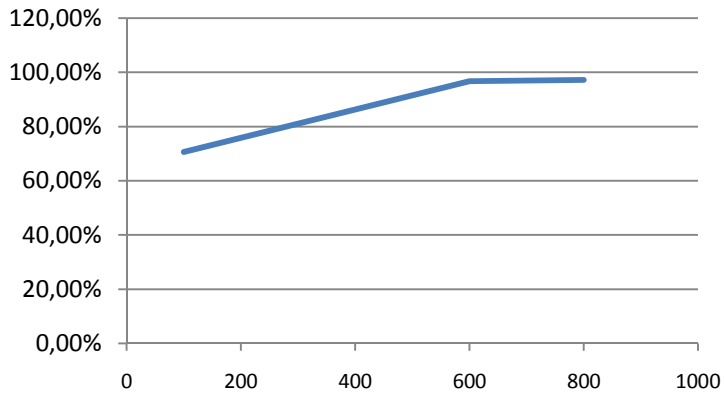


FIG IV.7: courbe d'expériences selon documents nombre de

Sample output to test PDF Combine only

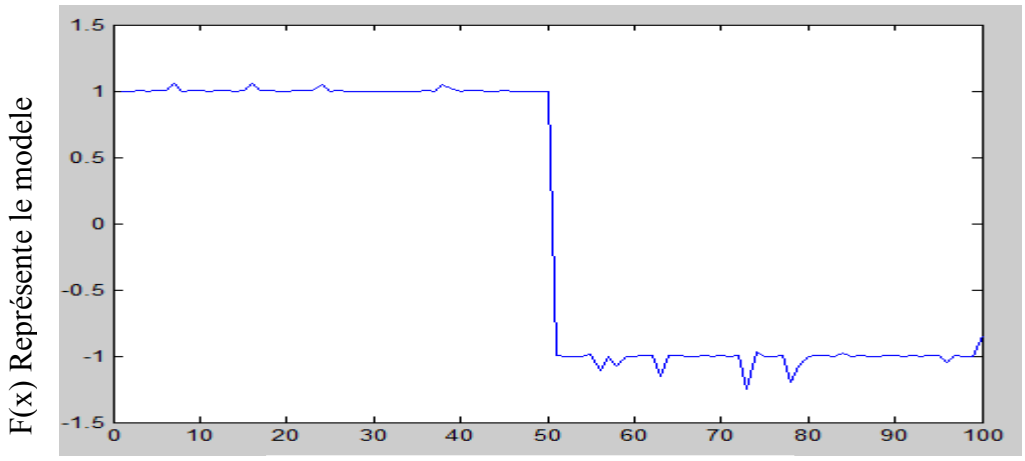


FIG IV.8: Résultats d'algorithmes pour 100 exemples d'apprentissage

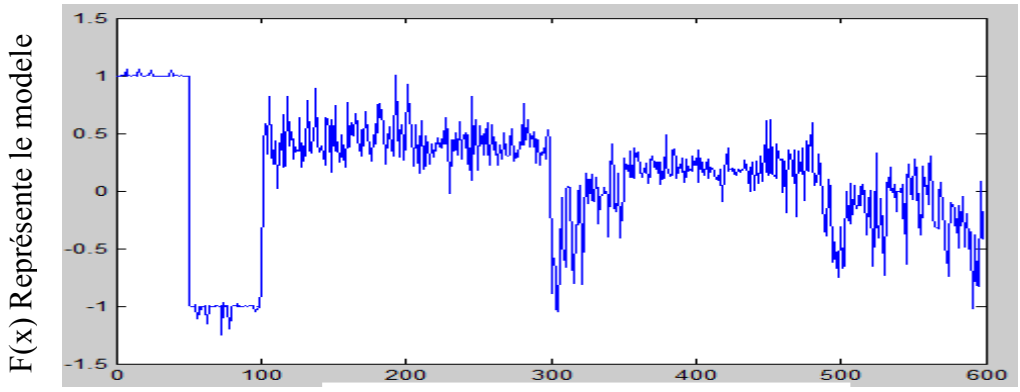


FIG IV.9: Résultats d'algorithmes pour exemples de test

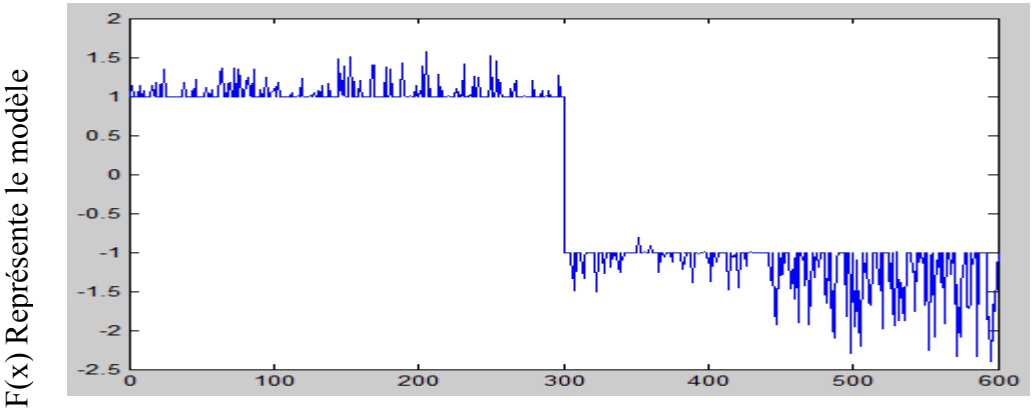


FIG IV.10: Résultats d'algorithmes pour 600 exemples d'apprentissage

Sample output to test PDF Combine only

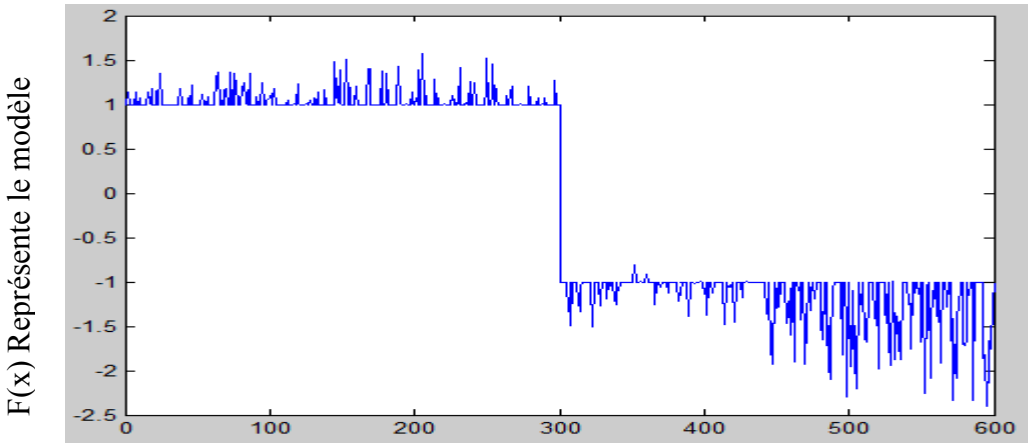


FIG IV.11: Résultats d'algorithmes pour exemples de test

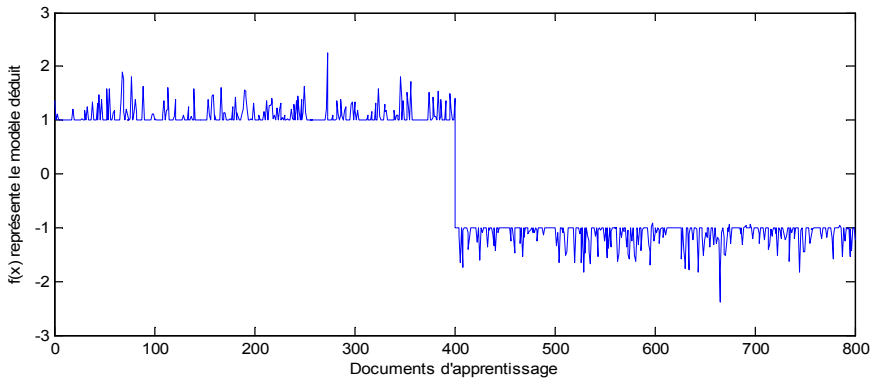


FIG IV.12: Résultats d'algorithmes pour 800 exemples d'apprentissage

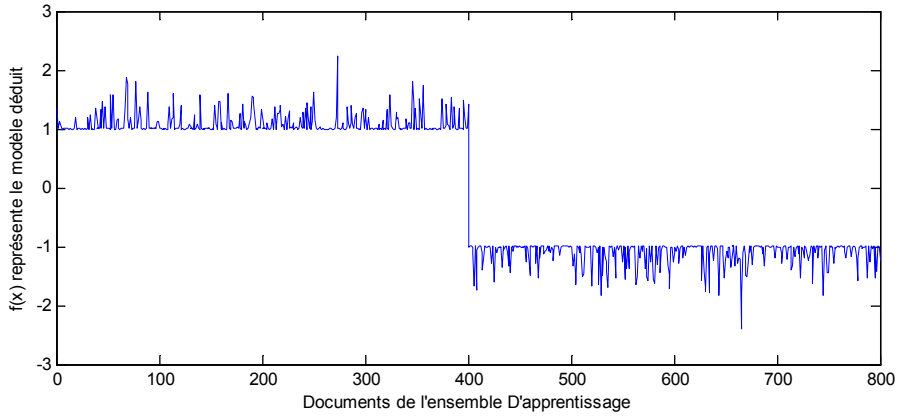


FIG IV.13: Résultats d'algorithmes pour exemples de test

Sample output to test PDF Combine only

IV.6)Argumentation et conclusion:

Nous allons interpréter ici les différents résultats parvenus des pratiques faites régulièrement et qui vous ont été montrées auparavant. Ces commentaires reposeront sur les épreuves des taux appartenant aux tableaux de résultats.

Pour **FIG IV.1** on a 50 document classer dans la catégorie -1 et 50 documents classer dans la catégorie +1

Nous avons commencé l'apprentissage avec un ensemble de 100 exemples dont 50 positifs et 50 négatifs, et qui sera augmenté pour chaque expérience, sous les mêmes valeurs des paramètres. Le tableau ci-dessus, montrent que le taux de performanc**e** augmente par l'augmentation de la taille du corpus .

Sample output to test PDF Combine only

CONCLUSION

Dans le cadre de cette thèse, nous proposons un classifieur de documents par apprentissage en utilisant la méthode « Supports Vectors Machine » (SVM) .Nous avons d'abord mettre en évidence les différents modèles de représentation des documents, précisément le modèle vectoriel en élabore les racines, les lemmes, les mots. Nous avons constaté ainsi que les racines représentent un choix très convenable puisqu'elles sont capables d'alléger significativement l'effet négatif du phénomène

Ensuite nous nous sommes penchés sur la question de l'efficacité de l'algorithme du « Supports Vectors Machine » (SVM) avec la classification automatique des documents et ca formulation qui devient un problème d'optimisation quadratique incalculable par la machine. Dans ce contexte plusieurs approches sont proposées et qui se reposent généralement sur le principe de la décomposition. En particulier, nous avons traité les algorithmes d'Osuna, SMO et SVM^{light}

Dans le domaine de l'apprentissage artificiel les pistes de la recherche sont nombreuses. L'accouplement des techniques de cette dernière avec celles du traitement automatique de la langue pousse les frontières de ces pistes encore plus loin. L'obtention d'une Master est le début d'une carrière consacrée à la recherche et l'avancement scientifique. Les travaux menés dans cette thèse ne sont qu'un point dedépart qui ouvre la voie envers d'autres perspectives qui méritent d'être explorées.

Sample output to test PDF Combine only

Résumé

L'apprentissage du support vector machine (SVM) conduit à un problème d'optimisation quadratique sous contraintes linéaires bornées. Malgré ce problème est claire, Il devient impossible, en termes de stockage mémoire et temps d'apprentissage, d'être résolu pour un nombre d'exemples d'apprentissage très élevé. Pour l'objectif de réduire le temps d'apprentissage, on propose ici un algorithme qui s'inspire de la méthode de décomposition proposé par Osuna dédié à l'optimisation des SVMs : il segmente le problème d'optimisation initial en sous problèmes calculable par la machine**.**

Mots clés : Représentation vectorielle, Classification, Apprentissage, Support Vector Machines (SVM), Optimisation quadratique, Décomposition.

Abstract

Training a support vector machine (SVM) leads to a quadratic optimization problem with bound constraints and one linear equality constraint. Despite the fact that this type of problem is well understood, there are many issues to be considered in designing an SVM learner. In particular, for large learning tasks with many training examples, off-the-shelf optimization techniques for general quadratic programs quickly become intractable in their memory and time requirements. Here we propose an algorithm based on the decomposition method proposed by Osuna dedicated to optimizing SVMs: it divides the original optimization problem into sub problems computable by the machine.

Keywords: Vector representation, Classification, Learning, Support Vector Machines (SVM), Qquadratic optimization, Decomposition.

Sample output to test PDF Combine only

Sample output to test PDF Combine only

Bibliographie

[Besançon, 2002] R.Besançon, « *Intégration de connaissances syntaxiques et sémantiques dans les représentations vectorielles de textes - Application au calcul de similarités sémantiques dans le cadre du modèle DSIR* », Thèse de doctorat - EPFL - Lausanne, 2001.

[Cortes et al, 1995] C. Cortes et V. Vapnik. « *Support vector networks*». Machine Learning, 20:1-25, 1995.

[Joachims, 1998] T. Joachims, « Making large-scale support vector machine learning practical ». In A. Smola B. Scholkopf, C. Burges, editor, Advances in Kernel Methods :Support Vector Machines. MIT Press, Cambridge, MA, 1998.

[Karush,1939] W.Karush,«*Minima of functions of several variables with inequalities as side constraints*».Master's thesis, Dept. of Mathematics, Univ. of Chicago, 1939.

[Kuhn et Tucker,1951] W. Kuhn et A. W. Tucker. « *Nonlinear programming »*. In Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probabilistics, pages 481–492, Berkeley, 1951.University of California Press.

[Mitchell, 1997] Mitchell, T. M. (1997). Machine Learning Computer Science. McGraw-Hill. New York.

[Osuna et al. 1997a] E.Osuna, R. Freund, and F. Girosi, « *Training Support Vector Machines: an Application to Face Detection*». Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97), New York, 1997 IEEE.

[Osuna et al, 1997b] E.Osuna, R.Freund, and F. Girosi. «*Support vector machines: Training and applications*». A.I. Memo 1602, MIT A. I. Lab., 1997.

[Plantié, 2006] M.Plantié, « *Extraction automatique de connaissances pour la décision multicritère* », Thèse de doctorat - ENSM- SAINT ETIENNE, 2006.

[Platt, 1998] C. Platt, « *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*». Technical Report MSR-TR-98-14, April (1998).

[Porter, 1980] M.F.Porter, « *An algorithm for suffix stripping* ». Program 14, pp 130-137. 1980.

[Quinlan, 1993] J. R. Quinlan, «*C4.5: Programs for Machine Learning*». Morgan Kaufmann, 1993.

[Quinlan, 1996] J. R. Quinlan, « *Bagging, boosting, and C4.5* ». In Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference, pages 725–730, Menlo Park, 1996.AAAI Press/MIT Press.

[Rad, 2003] RADWAN JALAM : « Apprentissage automatique et catégorisation de textes multilingues » Université Lumière Lyon2, Juin 2003.

Sample output to test PDF Combine only

Bibliographie

[Rosenblatt, 1958] F.Rosenblatt, «*The perceptron : A probabilistic model for information storage and organization in the brain*». Psychological Review, 65(6) :386–408, 1958.

[Salton , 1975a] G.Salton, « *A vector space model for automatic indexing*». Communicationsof the ACM, 18(11):613–620.

[Salton et al, 1975b] G.Salton , Yang, and Yu, «*A theory of term importance in automatic text analysis*». Journal of the American Society for Information Science and Technology, Volume 26(1) pages 33-44, 1975.

[Salton, 1981] G.Salton, «*A Blueprint for Automatic Indexing* ». SIGIR Forum, 16,(2). 1981.

[Salton, 1983] G.Salton, «*Introduction to modern information retrieval* ». MCGILL M. J., 1983.

[Salton et al, 1988] G.Salton and C.Buckley, Term-weighting «*Approaches in automatic text retrieval* ». Information Processing & Management, Volume 24, Issue 5, Pages 513-523 (1988).

[Salton et McGill, 1983] Salton, G. and McGill, M. J. 1986 Introduction to Modern Information Retrieval. McGraw-Hill, Inc.

[Sebastiani ,1999] F.Sebastiani, «*Machine learning in automated text categorisation*». Technical Report IEI-B4-31-1999, Consiglio Nazionale delle Ricerche, Italy, Pisa, IT, 1999.

[Sebastiani, 2002] F.Sebastiani, « *Machine learning in automated text categorisation*». ACM Computing surveys. 2002.

[Sim, 2005] SIMON RÉHEL : « Catégorisation automatique de textes et cooccurrence de mots provenant de documents non étiquetés » Mémoire présenté à la Faculté des études supérieures de l'Université Laval, Québec, Janvier 2005

[Vapnik, 1995] V. Vapnik, «*The Nature of Statistical Learning Theory*». Springer Verlag, New York, 1995.

[Zoutendijk, 1970] Zoutendijk, « *Methods of Feasible Directions: a Study in Linear and Non-linearProgramming* ». Elsevier, 1970.

Sample output to test PDF Combine only